

Head | Title | Body | Src | CSS | Table

许茂伟 马军 编著

内容特色

- 知识全面 囊括了HTML的所有元素和相应属性
- 实例经典 用实例演示了各个元素，这些例子可以直接应用到项目中
- 图文并貌 对每个实例都给出执行效果，方便学习
- 案例提高 最后给出了一个商业案例，告诉读者如何综合应用各种技术

本书看点

67个元素+**223**个属性+**363**个实例+**87**个习题+**6894**行代码=网页开发高手

光盘内容 DVD

- **342**个实例
- 赠送**485**页PPT文档
- **15**小时视频讲解
- **930**页电子书学习资料

售后服务

QQ群：21948169

Email：genwoxuebook@163.com

论 坛：rzchina.net

Blog：http://genwoxuehtml.blog.51cto.com

清华大学出版社

跟我学

跟我学 HTML+CSS

许茂伟 马军 编著

清华大学出版社
北 京

内 容 简 介

HTML 是目前最流行的网页制作语言。互联网中的网页大多数都是使用 HTML 格式展示在浏览者面前的。随着 Web 2.0 概念的提出，要求网页具有更好的扩展性和用户体验，这使得 CSS 样式表在网页设计中变得越来越重要。为了能让广大的网页制作者快速地掌握网页制作的技巧和方法，本书的第一部分以语法和实例相结合的形式详细讲解了 HTML 语言中各个元素及其属性的作用、语法和显示效果。第二部分从 CSS 基本概念开始，分别讲解了 CSS 盒模型和定位属性，CSS 控制各种元素显示的方法，CSS 布局页面的技巧等知识。最后一部分为了增强读者的实战能力，以个人博客制作的形式详细为读者演示了使用 HTML 和 CSS 制作页面的过程。为了便于理解，本书对所讲解的每个元素和属性，都做了实例演示。

为了方便读者学习，本书光盘中提供了丰富的内容，包括全书的多媒体视频演示、全书的电子教案、900 多页的电子资料以及书中讲解的源代码等内容。对于每章后面的习题，笔者都给了相应的解答，读者可以到 http://www.tupwk.com.cn/网站下载。

本书适合广大 Web 网站设计人员、网站设计的初学者、网站管理维护人员、大专院校学生和社会培训学生阅读，并可作为网站开发人员的参考手册。

图书在版编目(CIP)数据

跟我学 HTML+CSS/许茂伟，马军　编著．—北京：清华大学出版社，2010.9

(跟我学)

ISBN 978-7-302-23441-8

Ⅰ. 跟…　Ⅱ. ①许… ②马…　Ⅲ. ①超文本标记语言，HTML—主页制作—程序设计　②主页制作—软件工具，CSS　Ⅳ. ①TP312 ②TP393。092

中国版本图书馆 CIP 数据核字(2010)第 063804 号

责任编辑：胡辰浩(huchenhao@263.net)　袁建华

装帧设计：孔祥丰

责任校对：成凤进

责任印制：

出版发行：清华大学出版社　　**地　　址：**北京清华大学学研大厦

http://www.tup.com.cn　　**邮　　编：**100084

社　总　机：010-62770175　　**邮　　购：**010-62786544

投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn

质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn

印 刷 者：

装 订 者：

经　　销：全国新华书店

开　　本：185×260　**印　张：**30.25　**字　数：**716 千字

附光盘 1 张

版　　次：2010 年 9 月第 1 版　**印　次：**2009 年 9 月第 1 次印刷

印　　数：1~5000

定　　价：56.00 元

产品编号：



PREFACE

为了适应网络和网站制作技术的飞速发展，网页必须具有广泛的扩展性，代码必须简洁适用。目前有很多从事和即将从事网页制作和设计的人员，有很大一部分人无法正确理解和使用HTML语言的特点，而只能通过所见即所得的可视化开发工具制作网页。虽然可视化开发工具可以更加方便快捷地制作出页面，但是由于其本身的局限性，会导致代码的冗余等问题。了解HTML和CSS语言并结合相应的可视化开发工具，可以让代码的编写更加事半功倍。本书详细讲解了各种HTML语言及CSS属性的使用和布局的技巧，以及需要注意的问题等，能够较好地满足广大网页制作人员实际工作的需要。

本书作者根据自己多年的使用经验，将所有的HTML和CSS知识进行了筛选和整理，目的是让读者能够快速、全面、完整地掌握网页制作方法和技巧，系统掌握HTML和CSS的各种基本知识，并避免在部分不常用的代码上花费时间。本书讲解的HTML和CSS知识非常实用，是广大网页制作人员提高制作水平、完善知识结构、顺应网络技术发展的参考书。

本书的内容优势

本书第一部分详细讲解了HTML语言中各个元素的特点和各种属性的使用技巧，第二部分讲解了CSS的各种属性的作用，以及使用CSS美化网页的知识。最后一部分是实战中的应用技巧。在讲解每个知识点时，都结合实际制作的需要，使用简单的实例进行了演示。同时对实际应用中通常会发生的错误进行了分类讲解，便于读者系统全面地掌握各种技巧。

本书的特点主要体现在以下几个方面。

- 知识全面，内容充实。书中详细讲解了HTML语言中的所有元素和相应的属性，以及每个属性取值的知识。并将CSS中所有被浏览器支持的属性，按照应用分为几个大类。对每个分类中的各个属性都从作用、语法、使用方法、注意问题等几个方面进行了详细讲解。
- 遵循标准，注重规范。书中所有的代码和实例等内容，都遵循W3C发布的Web标准。不论CSS部分还是XHTML部分，代码的书写和属性的定义，都严格遵守相应的规范，为读者的学习提供了很好的范本。
- 条理清楚，注重应用。书中对HTML以及CSS的相关知识进行了筛选，摒弃了部分不实用和不规范的属性，让读者能够把握知识的重点，少走弯路。
- 叙述简洁，通俗易懂。书中对每个知识点的讲解都非常准确精炼，避免读者产生歧义。

本书的体例编排优势

为了方便读者学习，本书在写作方法上进行了专门的琢磨，主要包括如下几点。

- ❑ 实例经典，举一反三。书中对各个元素和属性都进行了实例演示，在每个实例中都演示了元素或者属性的典型应用，便于读者的理解。
- ❑ 图文并茂，重点突出。书中每个实例的演示效果，都用图片的方式展示出来，做到明确直观。
- ❑ 对程序代码进行编号，方便读者书盘结合，能轻易找到光盘中的源代码，从而提高学习效率。
- ❑ 每章最后配有习题，让读者总结提高，如果自己不能解答的话，我们还给出了答案，以加强学习。
- ❑ 分类十分细致，我们确保每个知识点都体现在目录，读者根据目录，就可以轻松找到需要的内容。

光盘内容

(1) 全书所有实例的源代码以及对应的素材文件。
(2) 15 小时视频讲解。
(3) 赠送 485 页 PPT 文档。
(4) 赠送 930 页电子资料。

本书的内容安排

本书分为 3 篇，共 20 章，从 HTML 使用的基本内容和概念讲起，循序渐进地讲解了 HTML 和 CSS 的相关知识、各种元素和属性的使用方法以及各种技巧等。其中每个部分的主要内容如下所示。

第一篇(第 1 章~第 10 章)HTML 的相关概念。

详细讲述了 HTML 的各种概念和相关内容。首先介绍了 HTML 的基础知识，接着详细讲解了 HTML 的页面基本元素、文本和段落元素、列表元素、图像元素、表格元素、超链接元素、表单元素、框架元素等知识。

第二篇(第 11 章~第 19 章)CSS 的相关概念。

首先讲述了 CSS 布局中的各种概念和相关内容。其中包括 CSS 的概念、结构和表现分离的原理和优点、XHTML 的基础知识、CSS 的基本语法和使用方式等。接着讲述了 CSS 盒模型和块元素的定位。包括 CSS 中盒模型的构成，块元素与内联元素的分类，以及使用各种属性控制块元素的位置和显示方式等。最后讲述了 CSS 控制元素显示效果和布局页面的方法。

第三篇(第 20 章)实战篇。

讲解了实际开发中，使用 HTML 和 CSS 代码的技巧。其中包括实际站点的建立、站点结构的规划、页面实例每个部分的制作过程等几个内容。通过本章的学习，可以最终运用可视化

的开发软件，结合本书中讲解的各种知识，完整地掌握页面制作的方法。

适合阅读本书的读者

- ❑ 专业的网页制作人员。
- ❑ 广大的网页设计爱好者。
- ❑ 网页程序代码编写人员。
- ❑ 专业的网页维护人员。
- ❑ 培训机构、高等院校及职业院校的学生。
- ❑ 从事网页代码优化的工作人员。
- ❑ 拥有个人站点的站长。

本书由洛阳理工学院的许茂伟、马军组织编写，其中许茂伟负责编写第 1 章~第 9 章，马军编写第 10~20 章，同时参与资料整理的有付京君、刘丹、张丹、文明、李刚、李里、杨丽、杨全德、杨明、武勇、段文睿、王安平、王文龙、肖运香、董方、郭军军、郭瑞、陈军、陈洁，在此一并表示感谢。

学习进度表

章　名	重点掌握内容	教学课时
第 1 章 HTML 的基本概念	1. 什么是 HTML 2. HTML 与网页的关系 3. 用什么来编写和开发 HTML	1 学时
第 2 章 HTML 的语法基础	1. HTML 页面结构 2. 元素和属性的写法 3. 文档类型	1 学时
第 3 章 页面基本元素	1. 页面基础元素 2. 页面头部元素 3. 页面头部相关元素 4. 页面主体元素	2 学时
第 4 章 文本和段落元素	1. 层元素 2. 标题元素 3. 段落元素 4. 文本的间隔和布局 5. 特殊的文本元素	2 学时
第 5 章 列表元素	1. 无序列表元素 2. 有序列表元素 3. 列表条目元素 4. 定义列表元素	1 学时
第 6 章 图像元素	1. 图像元素 2. 图像的格式	1 学时
第 7 章 表格元素	1. 表格元素的结构 2. <table>元素的属性 3. <tr>元素的属性 4. <td>元素的属性 5. 表格中使用的其他元素	3 学时
第 8 章 链接元素	1. 链接和路径 2. 链接元素 3. 图像链接	2 学时

(续表)

章　名	重点掌握内容	教学课时
第 9 章 表单元素	1. 表单元素 2. 表单控件 3. <input>元素的属性	2 学时
第 10 章 框架元素	1. 框架集元素 2. 框架内容元素 3. 不支持框架元素 4. 内联框架元素	2 学时
第 11 章 CSS 的概念	1. 什么是 CSS 2. CSS 与网页显示效果的关系 3. 使用 CSS 的方法	1 学时
第 12 章 CSS 的语法	1. 选择符 2. 属性 3. 值 4. 块元素和内联元素 5. 应用样式的优先级	1 学时
第 13 章 CSS 控制文本的显示	1. 控制字体的显示 2. 控制文本的显示	1 学时
第 14 章 CSS 控制列表元素的显示	1. 控制列表元素的显示 2. 列表元素的使用和嵌套	1 学时
第 15 章　CSS 控制表格元素的显示	1. 控制表格元素的显示 2. 单元格的制约关系	2 学时
第 16 章　CSS 控制元素的大小	1. 盒模型的概念 2. 元素内容的大小 3. 元素的背景 4. 元素的补白 5. 元素的边框 6. 元素的边界 7. 嵌套元素的大小和距离	4 学时
第 17 章　CSS 控制元素的定位	1. 元素的定位 2. 绝对定位 3. 相对定位 4. 定位元素的重叠	3 学时

(续表)

章　　名	重点掌握内容	教 学 课 时
第 18 章　CSS 控制元素的布局	1. 元素的浮动 2. 浮动的清除 3. 内容的剪切 4. 溢出内容的控制 5. 元素的显示方式 6. 元素的可视性	4 学时
第 19 章　CSS 控制其他元素的显示	1. 控制滚条的显示 2. 控制光标的显示 3. 控制元素的缩放 4. 控制链接的显示	2 学时
第 20 章　制作个人博客页面	1. 制作页面前的准备工作 2. 规划站点文件夹 3. 定义基本的样式 4. 制作页面各个部分	4 学时



目录

C O N T E N T S

Chapter 1

HTML的基本概念

随着网络的不断普及，网页已经被大多数人所熟悉。大家每天浏览的各大站点都是由一页一页的网页构成的。但是这些页面是怎样搭建起来的呢？又是怎样显示的呢？其实网页是由一种简单的标记语言 HTML 构成的。本章将讲解 HTML 的基本概念，以及开发 HTML 所使用的软件的相关知识。

本章主要内容有:

◎ HTML 的概念。
◎ HTML 和网页的关系。
◎ HTML 的编译环境。
◎ HTML 的示例。

1.1 什么是 HTML

HTML 的英文全称是 Hyper Text Markup Language，直译为超文本标记语言，它是全球广域网上描述网页内容和外观的标准。大家在浏览网页时可能注意到很多页面的后缀名为.html，事实上，HTML 是一种因特网上较常见的网页制作标注性语言，它并不能算做一种程序语言，因为它缺少语言所应有的特征。HTML 通过 IE 等浏览器的翻译，将网页中所要呈现的内容展现在用户眼前。如何查看 HTML 代码呢？以大家最常用的 IE 浏览器为例，基本的操作步骤如下所示。

(1) 双击 IE 浏览器图标，打开 IE 浏览器。

(2) 在地址栏输入需要打开的网址，按 Enter 键打开相应的页面，如图 1-1 所示。

(3) 在浏览器中执行“查看”|“源文件”命令，打开源文件显示窗口，通常为一个文本文档，如图 1-2 所示。

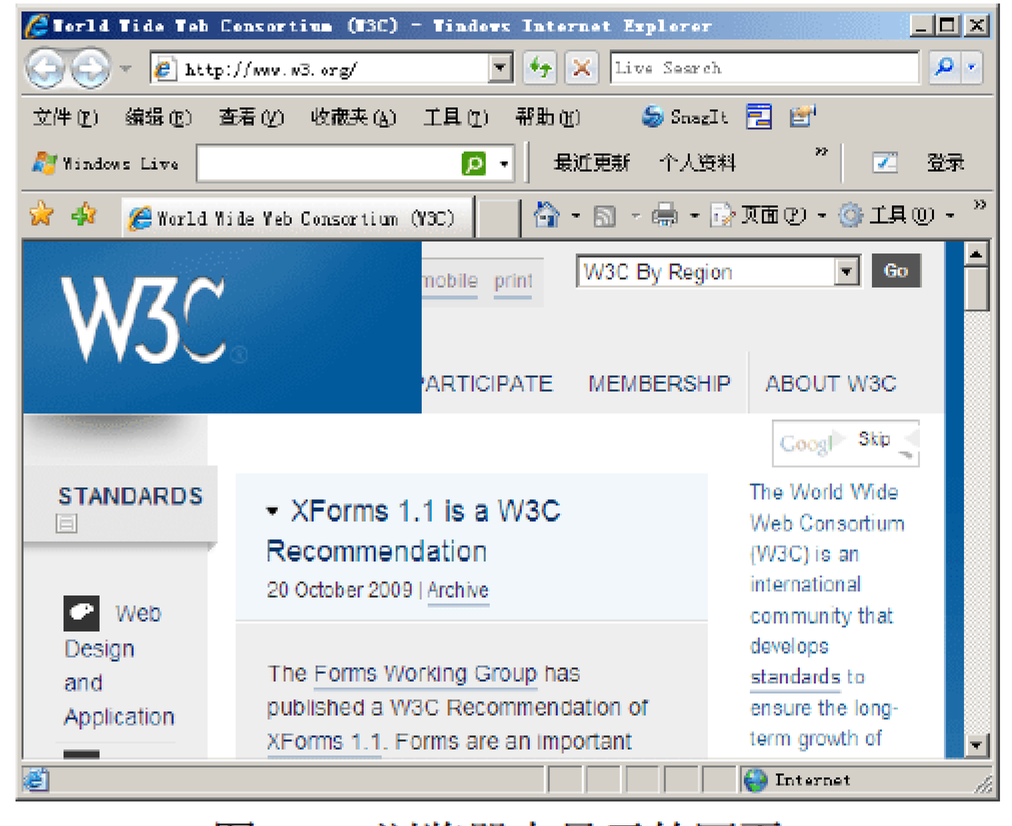


图 1-1　浏览器中显示的网页

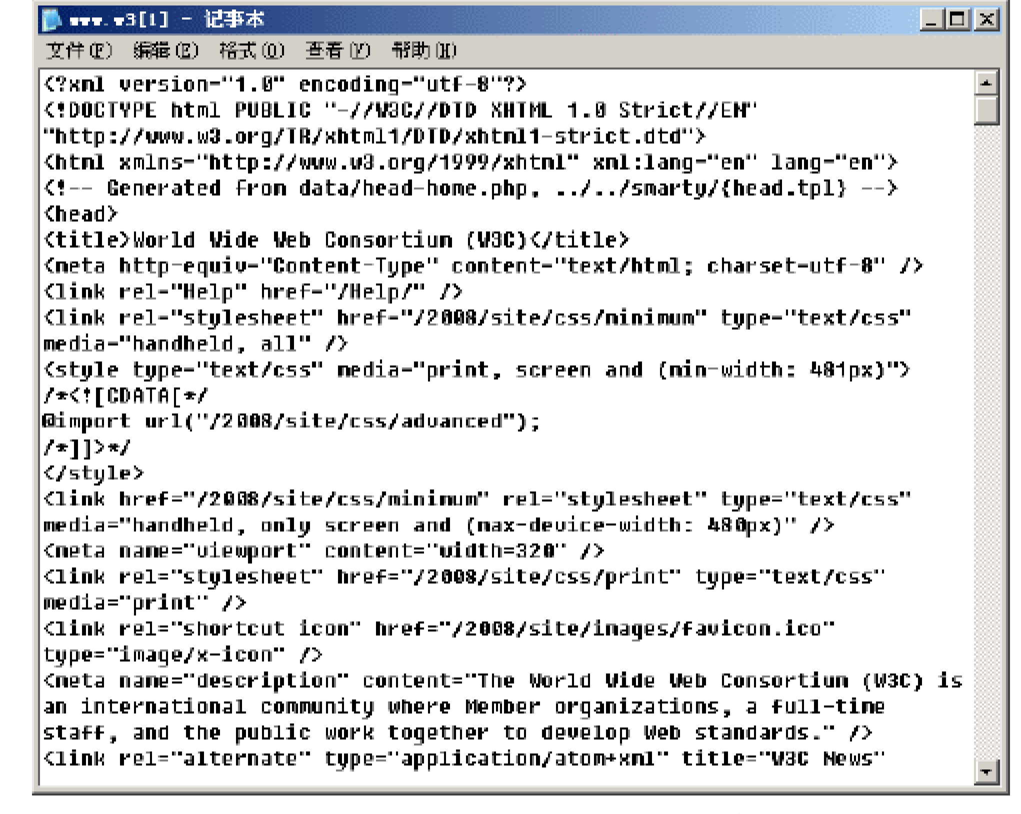


图 1-2　页面的 HTML 代码

从图 1-1 和图 1-2 可以看出，通常所看到的五彩缤纷的网页，其实都是由一系列的 HTML 代码构建起来的，掌握了如何编写 HTML 代码，就可以随心所欲地制作网页了。

1.2 HTML 的发展历史

HTML 的雏形是由蒂姆·本尼斯李创建的，用来解决科学家们共享网络信息的问题。最初的 HTML 语言以文本格式为基础，可以用任何编辑器和文字处理器来为网络创建或转换文本，仅有不多的几个标签。网络从此迅猛发展，人人都开始在网上发布信息。很快人们就开始琢磨在网上放置图像和图标。

1993 年，NCSA 推出了 Mosaic，也就是第一个图文浏览器，从此 Web 开始迅速地发展起来。HTML 语言也不断产生新型、功能强大且生动有趣的标签形式，如<background>、<frame>、<font>和<blink>等。

但是此时，出现了许多不同的 HTML 版本，而只有设计者和用户共有的 HTML 部分才可以正确显示。因此在这段时间，W3C 都在激烈争论名叫 HTML 3 的新技术，该文件概括了所有全新的特性但没有任何技术支持。出于这种混乱局面的考虑，在 1996 年 W3C 的 HTML Working Group 组织编写了新的规范，从此 HTML 3.2 开始发展，它更接近于现实的目标，即提供给内容商和浏览器发展商在研究工作中一个公允的参考标准。

到现在为止，HTML 已经发展到了比较成熟的 HTML 4.0 版本，在这个版本下的语言中，规范更加统一，浏览器之间的兼容性也更加完好。

HTML 包含了一对打开和关闭的标记，在当中并有属性和值。标记描述了每个在网页上的组件，如文本段落、表格或图像。在本书中，主要讲解 HTML 语言各种属性和应用。

1.3 一个简单的 HTML 示例

一个完整的 HTML 文件包括标题、段落、列表、表格以及各种嵌入对象，这些对象统称为 HTML 元素，在 HTML 中使用标签来分割并描述这些元素。实际上可以说，HTML 文件就是由各种 HTML 元素和标签组成的。

一个 HTML 文件的基本结构如下：

```
<html>              文件开始标记
<head>              文件头开始的标记
……                  文件头的内容
</head>             文件头结束的标记
<body>              文件主体开始的标记
……                  文件主体的内容
</body>             文件主体结束的标记
</html>             文件结束标记
```

从上面的代码结构可以看出，在 HTML 文件中，所有的标记都是相对应的，开头标记为< >，结束标记为</ >，在这两个标记中间可添加属性、数值、嵌套结构等各种类型的内容。

下面是一个简单的 HTML 示例的代码，如下所示。

例程 1-1 html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
```

```
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>无标题文档</title>
</head>
<body>
这是一个 HTML 的展示示例
</body>
</html>
```

在 Dreamweaver 中运行这段代码，就能看到常见的网页格式的显示页面了。效果如图 1-3 所示。

从图 1-3 可以看到<title></title>之间的内容显示为页面的名称，而<body></body>之间的部分会展示在页面内容中。

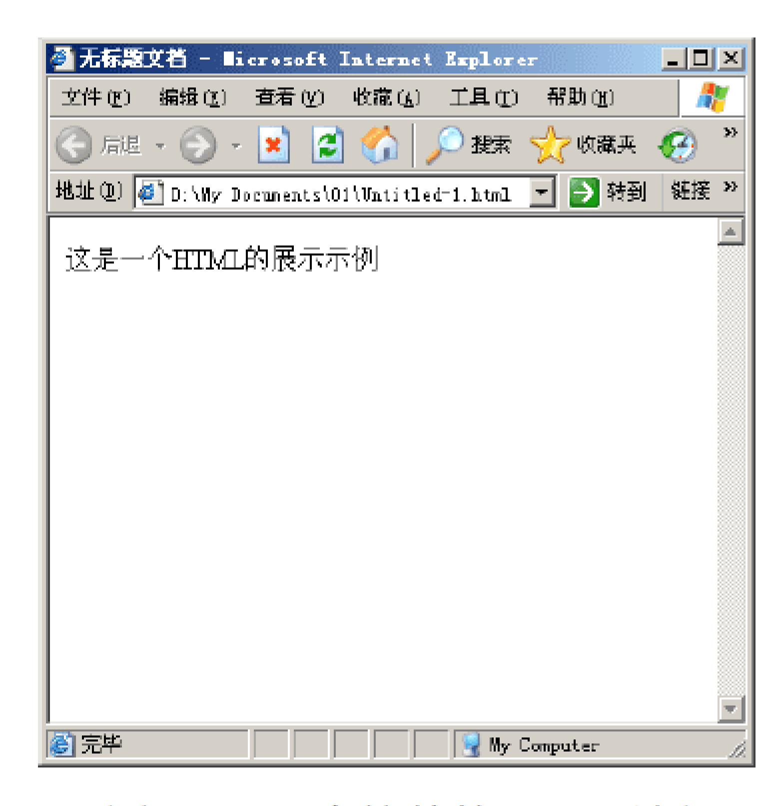


图 1-3　一个简单的 html 示例



1.4 HTML 与网页的关系

平时所浏览的 HTML 网页中含有文本段落、表格和图像等。下面是一个含有文本和图片的 HTML 示例，包括了 HTML 文档中必须使用的部分元素。其具体代码如下所示。

例程 1-2　html.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>有图像的 HTML 页面</title>
</head>
<body>
 <h3>西湖十景之雷峰塔</h3>
 雷峰塔简介<br>雷峰塔是杭州西湖的著名景点之一，我们熟悉的《白蛇传》中的故事就发生在这里，据说白娘子就是被压在这下面的，这里也是著名的西湖十景之一雷峰夕照。<br>
 <center>
 <img src=" pic.jpg">
 </center>
  </body>
  </html>
```

其代码运行后的显示效果，如图 1-4 所示。

对照代码和网页可以发现，HTML 的代码并不直接显示在网页上，而是通过某种对应的关系以网页的形式显示出来。HTML 代码的作用就是建立起网页内容与最终效果之间的关系，并通过对这些关系的调整达到构建和美化内容的目的。平时所看到的网页当然还要比这个复杂得多，在以后的学习中不断总结，就能做出精美的网页了。



图 1-4　简单的 XHTML 示例

1.5 用什么来编写和开发 HTML

Dreamweaver 是最常见的编写 HTML 的工具，使用这个工具可以随心所欲地编写代码、设计网站网页以及进行高级开发。无论是喜欢手写 HTML 代码还是习惯于可视化环境，Dreamweaver 都能提供方便快捷、功能强大的工具。在易用、创新、规范等优点的基础上，Dreamweaver 还拥有更先进的网页布局和设计环境以及更为强大的代码编辑功能等卓越特性。

Dreamweaver 的操作界面主要由以下几个部分组成：标题栏、菜单栏、插入栏、文档工具栏、文档窗口、属性窗口及多个浮动面板组成，如图 1-5 所示。

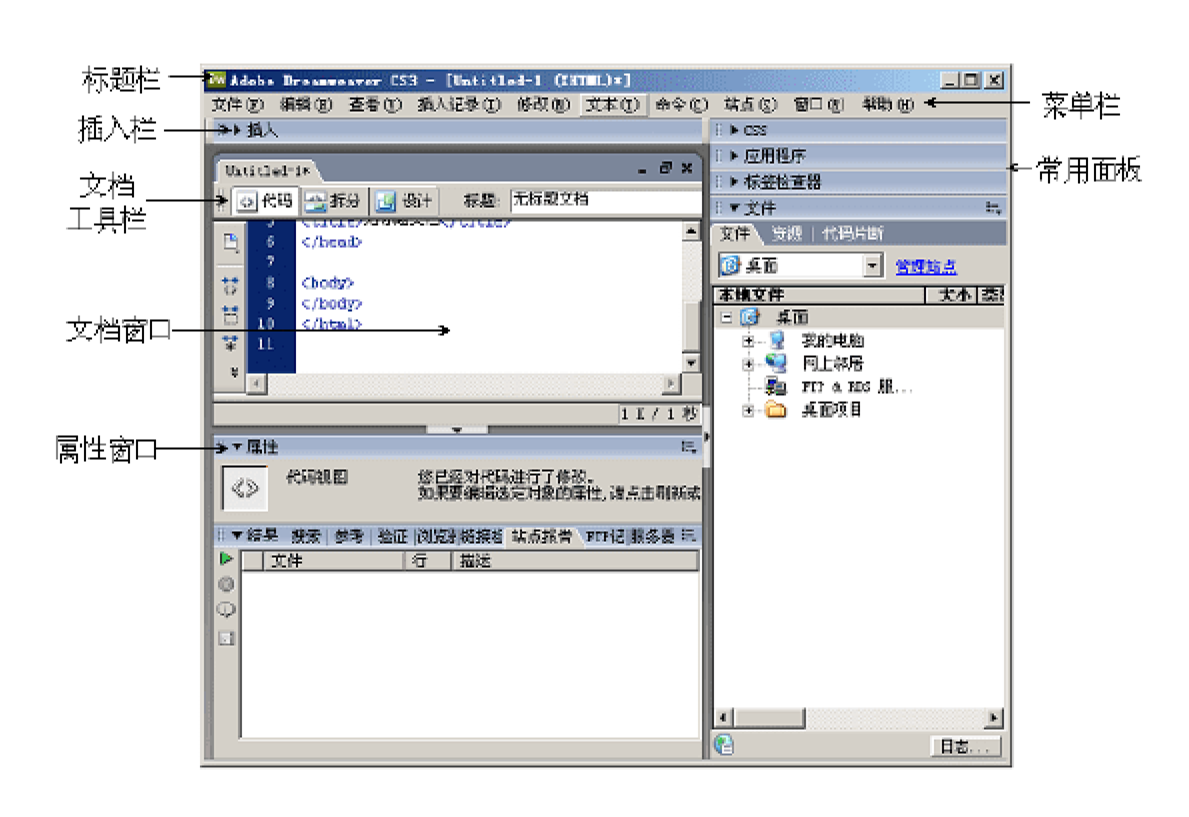


图 1-5　Dreamweaver 的界面布局

下面来看看 Dreamweaver 中各个部分的具体功能。

1.5.1　标题栏

标题栏主要包括 Dreamweaver 标记、应用程序的名称、当前正在编辑文档的标题和名称等信息，还包括最小化按钮、最大化按钮以及关闭按钮。

单击 Dreamweaver 标记 可以打开系统菜单。在 Dreamweaver 标记后面显示程序名称，之后的中括号“[ ]”内是当前打开的文档的标题，小括号“( )”内是该文件的名称。每次新建一个文档，Dreamweaver 都会自动将文档标题命名为“无标题文档”，文件名定义为 Untitled-x。其中，文档的标题和文档的文件名称是不同的概念。文件的标题通常在文档中的<title>和</title>标记中，是在用浏览器打开文档时显示在浏览器窗口的标题栏上的名称，而文件的名称则是文档存储在磁盘上的文件名。

1.5.2　插入栏

对于一些经常使用的操作，如从菜单项中选择还是很不方便。插入栏是 Dreamweaver 提供的一些菜单命令的快捷方法，其按钮一般都和菜单中的命令相对应，运用插入栏可以更方便快

捷。插入栏集成了多种网页元素，包括图像、文字等，默认的插入栏中的各种选项是隐藏起来的，如图 1-6 所示。



图 1-6　插入栏

单击插入栏左边的向下箭头▼，可以选择不同的工具组，包括布局、表单、文本等，如图 1-7 所示。如果选中“显示为制表符”选项则以传统的方式显示插入栏，如图 1-8 所示。

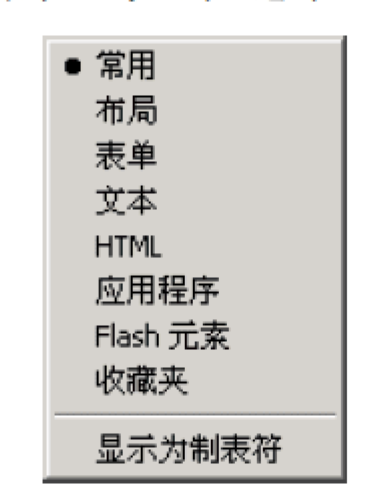



图 1-7　工具组菜单

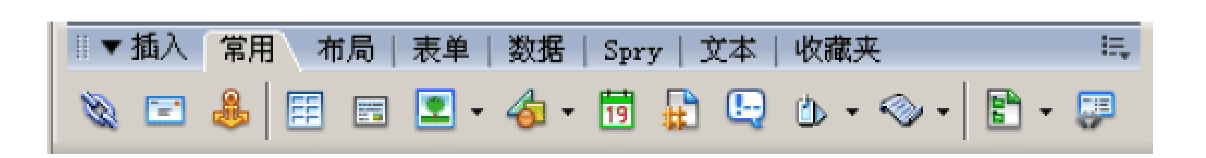


图 1-8　传统方式的插入栏

“收藏夹”是用户自定义的组，在这里用户可以创建自己常用的按钮。

1.5.3　文档工具栏

“文档”工具栏包含按钮和弹出式菜单，它们提供各种“文档”窗口视图(如“设计”视图和“代码”视图)、各种查看选项和一些常用操作(如浏览器中预览)，如图 1-9 所示。

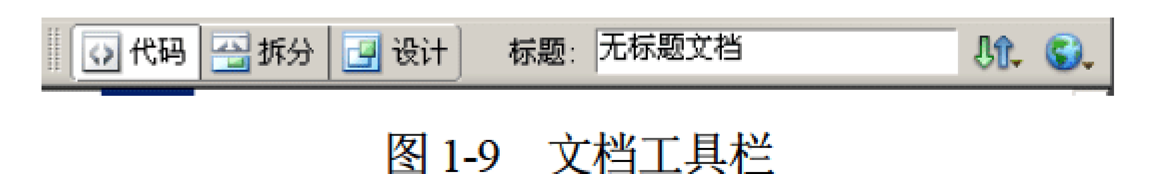


图 1-9　文档工具栏

要显示文档工具栏，可以单击“查看”|“工具栏”|“文档”菜单命令。文档工具栏中各个图标按钮的功能分别如下所示。

- “显示代码视图”：显示代码窗口。
- “显示代码和设计视图”：显示代码和设计窗口。
- “显示设计视图”：显示设计窗口。
- 标题: 无标题文档 “标题”：用来设置或修改文档的标题。
- “文件管理”：单击该按钮，通过这里来实现消除只读属性、获取、取出、上传、存回、撤销取出、设计注意以及站点定位等功能。
- “在浏览器中预览/调试”：单击该按钮在定义好的浏览器中预览或调试，或是编辑浏览器列表。

1.5.4　实例练习

在 Dreamweaver 中直接书写 HTML 的代码，通过一个 HTML 小实例进行说明。

第一步：编写 HTML 代码。

(1) 启动 Dreamweaver 程序，就会看到 Dreamweaver 的工作界面，如图 1-10 所示。



图 1-10 Dreamweaver 的工作界面

(2) 选择“文件”|“新建”命令，在打开的对话框中选择 HTML 选项，或者直接单击界面新建栏目下的 HTML 选项，都可以以代码视图的形式打开一个新的 HTML 文件，如图 1-11 所示。

(3) 在代码中<title>与</title>标签之间的内容就是新建 HTML 文件的标题，也就是将要在浏览器的标题栏中显示的页面标题，这里将其更改为“这是一个网页”。

(4) 在<body>与</body>标签之间添加主体内容的代码，如下：

```
<br>欢迎学习 HTML 语言！</br>
<br>我们愿做您最忠实的良师益友！</br>
<br>让我们伴您一起走入 HTML 的世界吧！！</br>
```

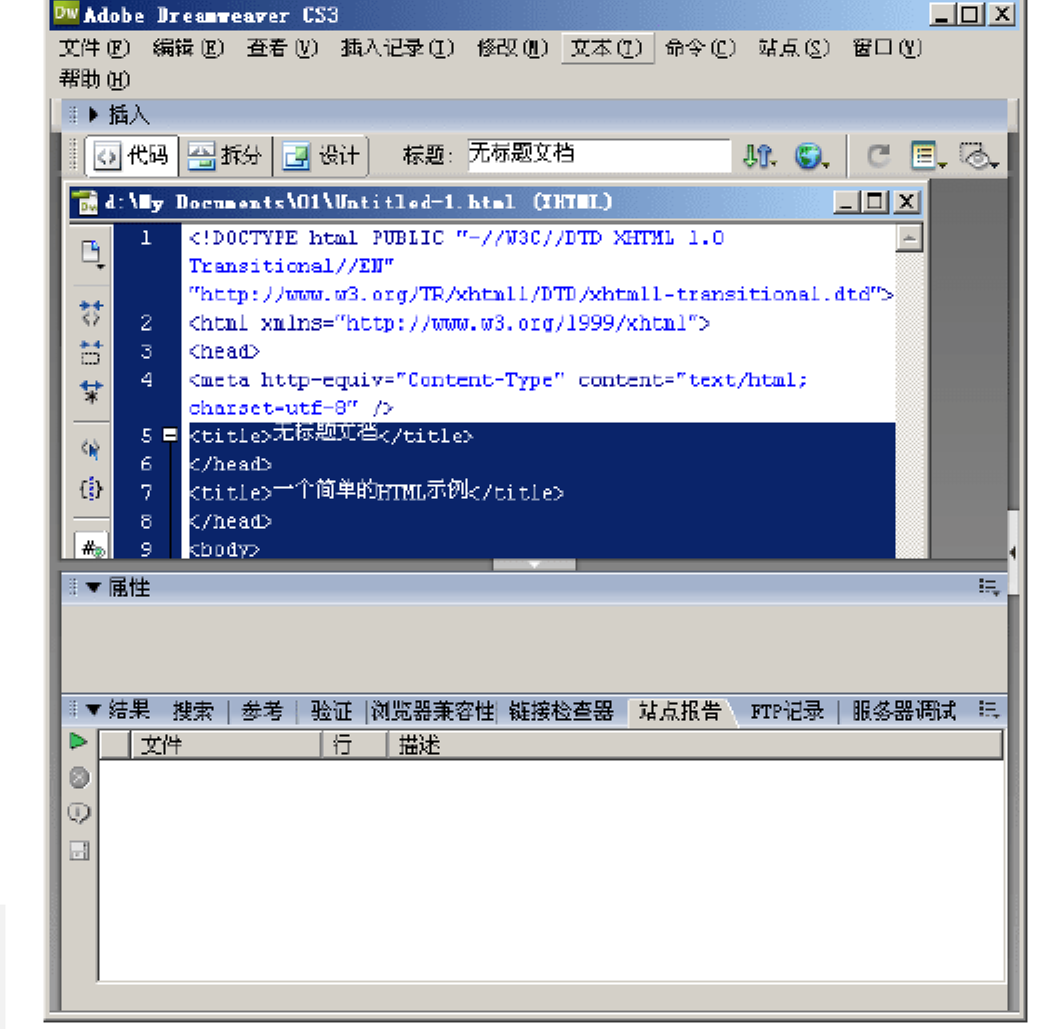


图 1-11 Dreamweaver 的代码视图

这样一个最基本的 HTML 文件就编写完成了。

第二步：保存并查看 HTML 文件。

(1) 选择“文件”|“保存”菜单命令，打开如图 1-12 所示的“另存为”对话框。

(2) 在“保存在”后面的下拉列表中选择存盘的位置，在“文件名”后面的文本框中输入文件的名称，设置文件的保存类型为“HTML 文档”，单击“保存”按钮完成文件的保存。

(3) 双击保存的文件，可以在浏览器中看到文件的效果，如图 1-13 所示。

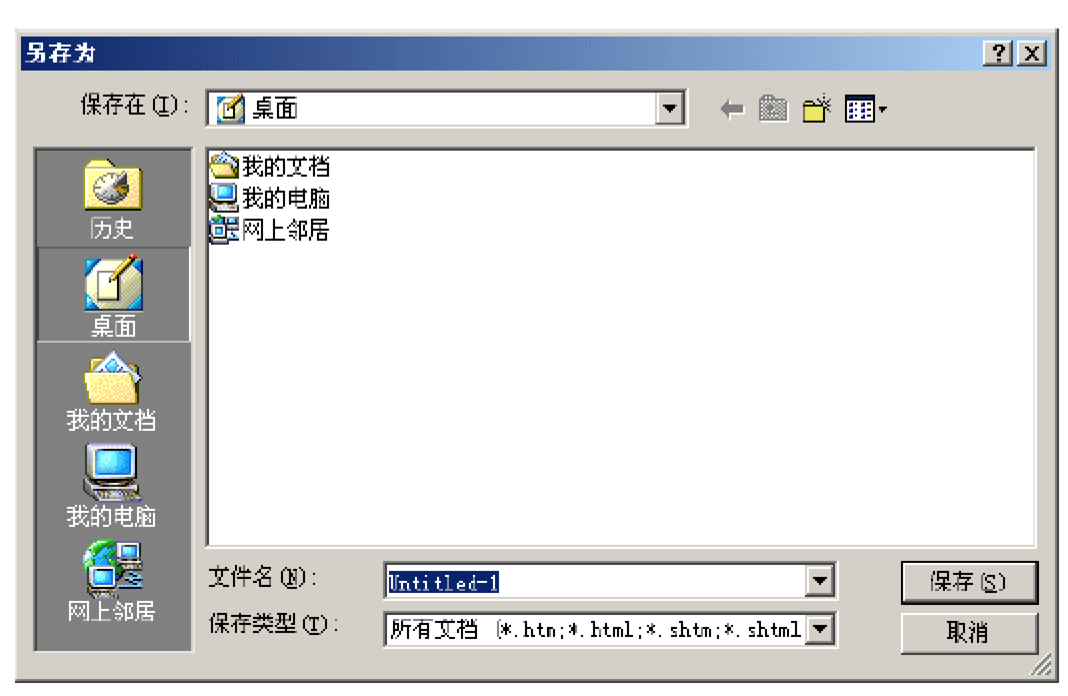



图 1-12　“另存为”对话框

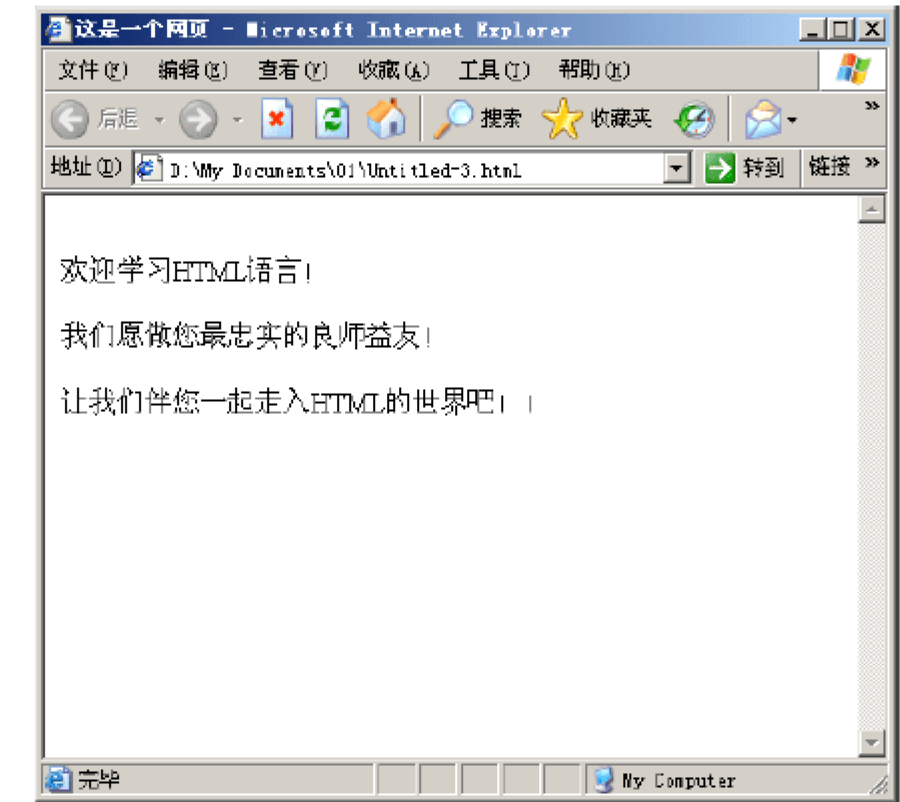


图 1-13　保存好后的显示效果

1.6 本章习题

1．HTML 编写的页面要在什么环境下才能显示？

2．使用 Dreamweaver 软件制作内容为“进入 HTML 的世界”，文档名称为“index”，标题为“HTML 练习”的页面。



HTML的语法基础

本章主要讲解关于 HTML 的基础知识。内容包括 HTML 页面结构、元素的书写格式和属性、HTML 代码规范、选择文档类型、定义名字空间等几个内容。

本章主要内容有:

◎ 重点掌握 HTML 页面结构、HTML 代码规范和文档类型选择的相关知识。

◎ 养成良好的程序书写习惯。

◎ 名字空间的含义。

2.1 HTML 页面结构

首先看一个最简单的 HTML 页面实例。其代码如下所示。

例程 2-1 xhtml- example.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title> HTML 页面结构</title>
</head>
<body>
这里是页面的主体
</body>
</html>
```

在这段代码中，包含了一个 HTML 页面必需的页面结构部分。其具体结构如下所示。

首先是文档类型声明部分，由<!DOCTYPE>元素定义。其对应的页面代码如下所示。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

<html>元素是 XHTML 文档中必须使用的元素，所有的文档内容(包括文档头部内容和文档主体内容)都要包含在<html>元素之中。<html>元素的语法结构如下所示。

```
<html>……</html>
```



说明

<html>和结束标签</html>一起构成一个完整的<html>元素。其包含的内容要写在起始和结束标签之间。

名字空间是<html>元素的一个属性，写在<html>元素起始标签里面。其在页面中的相应代码如下所示。

```
<html xmlns="http://www.w3.org/1999/xhtml">
```

名字空间属性用 xmlns 来表示，用来定义识别页面标签的网址(关于名字空间的详细内容，请参看 2.4 节)。

网页头部元素<head>，也是 XHTML 文档中必须使用的元素。其作用是定义页面头部的信息，其中可以包含标题元素、<meta>元素等。<head>元素的语法结构如下所示。

```
<head>……</head>
```

说明

<head>元素所包含的内容，不会显示在浏览器的窗口中。但是部分内容会显示在浏览器的其他位置(关于页面头部所使用的元素，请参看第 3 章)。

页面标题元素<title>用来定义页面的标题。其语法结构如下所示。

```
<title> ……</title>
```

说明

页面标题中包含的文本，在页面发布时，会显示在浏览器的标题栏中(关于页面标题的详细内容，请参看第 3 章)。

页面主体元素<body>用来定义页面所要显示的内容。页面的信息主要通过页面主体来传递。在<body>元素中，可以包含所有页面元素。<body>元素的语法结构如下所示。

```
<body>……</body>
```



说明

在制作页面时，经常要在<body>元素中定义相关属性，用来控制页面的显示效果。

定义了以上几个元素后，便构成了一个完整的 HTML 页面。而且以上所有元素，都是 HTML 页面所必须具有的基本元素。此时页面在 IE 浏览器中的显示效果，如图 2-1 所示。

注意

在本书中，如没有特意说明，代码运行后的显示效果，均为在 IE 浏览器中的显示效果(版本为 6.0)。

一个完整的 HTML 页面，必须包含以上几个元素。关于页面结构中包含的其他元素和属性，将在以后的章节中详细介绍。

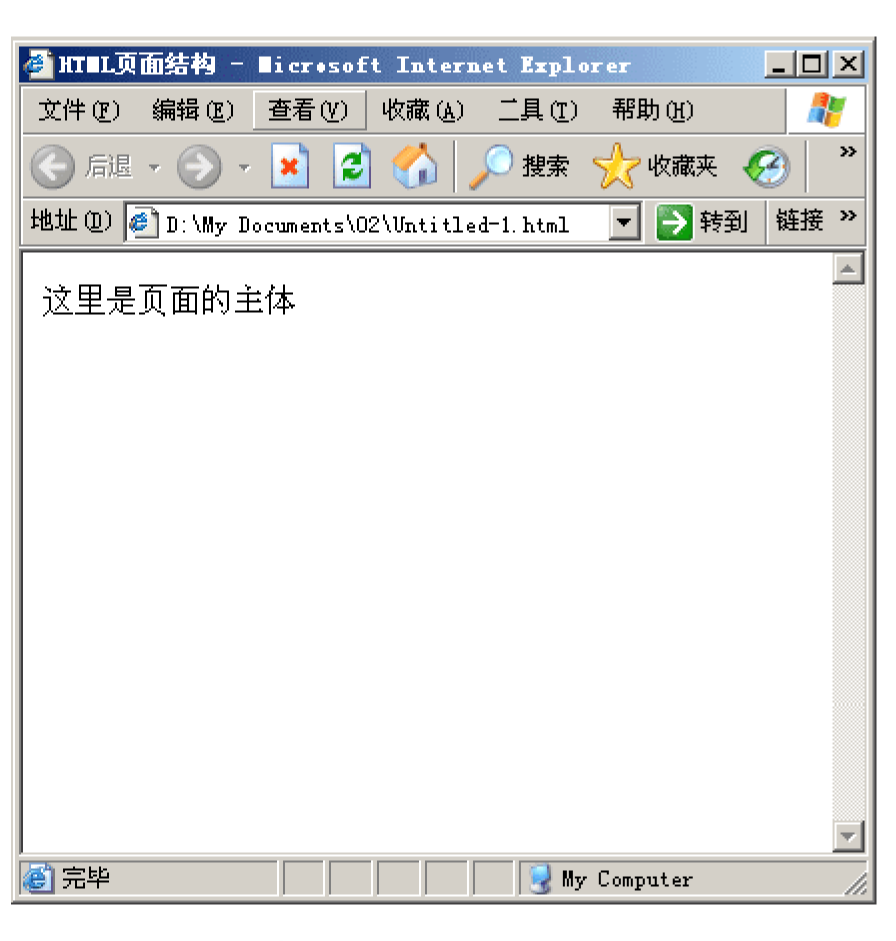


图 2-1 简单的 XHTML 页面的显示效果

2.2 元素和属性的写法

元素和属性是 HTML 的构成基础，每个元素在 HTML 页面中构成了不同的部分，可以将整个页面划分成不同的结构。而每个元素中的属性可以用来定义元素的显示效果，也可以用来标记元素，便于元素与其他的内容关联。熟悉各种 HTML 元素及其属性的功能是制作 HTML 页面的必备知识。

2.2.1 元素的书写格式

在确定了页面结构之后，就可以在结构中添加其他的页面元素了。其他页面元素从表现形式上看，可以分为两类：可以包含内容的元素、自封闭的元素。下面分别介绍这两种元素。

1. 可以包含内容的元素

可以包含内容的元素，其表现形式如下所示。

```
<元素名称>……</元素名称>
```

说明

可以包含内容的元素，其完整的结构由两部分组成，即元素的起始标签<元素名称>和元素的结束标签</元素名称>。

下面是一个使用含有内容元素的实例，其代码如下所示。

```
<title>……</title>
```

2. 自封闭的元素

自封闭的元素，是指起始和结束使用同一个标签的元素。其表现形式如下所示。

```
<元素名称 />
```

说明

在自封闭的元素中，采用“元素名称”之后添加英文“空格”和“/”的格式来结束元素。所以元素中不能包含内容。

2.2.2 元素属性的书写格式

属性是包含在元素之中，用来定义元素各种显示效果和触发相应行为的部分。其语法结构如下所示。

```
<元素名称 属性="属性值">……</元素名称>
```

说明

属性必须添加在元素的起始标签之中，用“空格”来分割属性与元素名称。属性值必须包含在英文的“引号”之中。可以为一个元素定义几个属性，其中每个属性之间用“空格”进行分隔。

属性的作用是用来设置元素的外观和行为。下面是一个在元素中使用属性的实例，其代码如下所示。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>赤壁怀古</title>
</head>
<body bgcolor="#00FFCC">
 大江东去浪淘尽，千古风流人物。
</body>
</html>
```

说明

代码中，在<body>元素中添加了 bgcolor 属性，其属性值为#00FFCC。#00FFCC 显示效果为浅绿色。bgcolor 属性的含义是背景颜色。

使用 bgcolor 属性后，页面的显示效果如图 2-2 所示。作为参照，取消 bgcolor 属性后，页面的显示效果如图 2-3 所示。

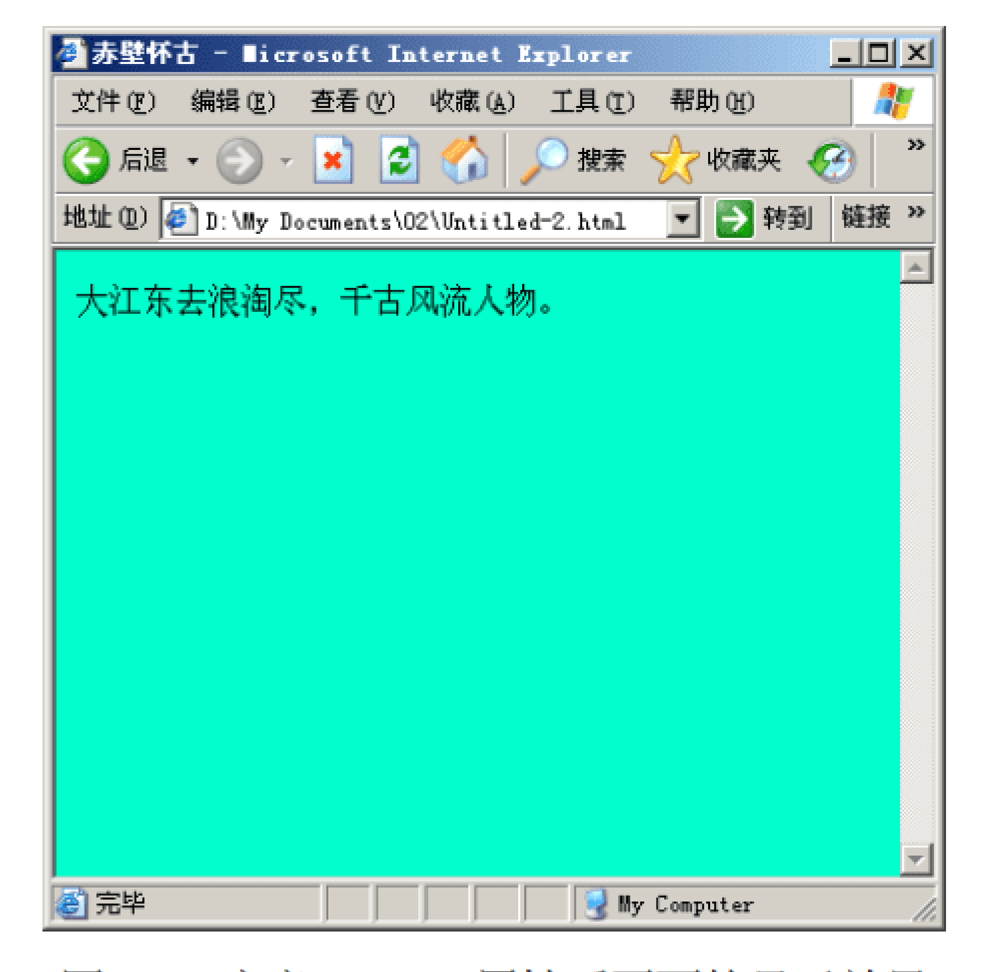


图 2-2 定义 bgcolor 属性后页面的显示效果



图 2-3 取消 bgcolor 属性后页面的显示效果

对比图 2-2 和图 2-3 可以看出，通过定义属性，可以更改元素默认的表现形式。一般每个元素都含有多个不同的属性。按照属性的作用，可以大致将属性分为以下几类。

- 表现属性用来控制元素表现效果，如元素的高度、宽度、边框、文本排列方式、语言等属性。
- 事件属性用来添加行为，一般要结合 JavaScript 等脚本或者程序来实现，如 onMouseOver 属性、onLoad 属性等。
- 标记属性用来标记某个元素，如 id 属性、name 属性等。
- 级联样式属性是指使用级联样式表，如 style 属性、class 属性等。
- 其他属性指以上几类属性以外的属性，如 type 属性、value 属性等。

关于元素属性的详细内容，将在讲解每个元素使用方法时具体介绍。

2.2.3 HTML 的语法规范

了解上面的内容后，用户在使用 HTML 语言进行网页制作时，必须要遵循一定的语法规范。下面进行详细讲解，其中具体内容可以分为以下几点。

1. 区分大小写

XHTML 对大小写是敏感的，在 XHTML 文档中，使用相同字母大写和小写定义的元素是不同的。例如，<h>和<H>表示的是不同的元素。

说明

在 HTML 中，规定要使用小写字母来定义页面中所有的元素和属性。包括 CSS 样式表中的属性等也要使用小写字母。

2. 正确嵌套所有元素

HTML 中，当元素进行嵌套时，必须按照打开元素的顺序进行关闭。正确嵌套元素的代码示例如下。

```
<ul><li></li></ul>
```

错误的嵌套元素的代码示例如下所示：

```
<ul><li></ul></li>
```

HTML 中还有一些严格强制执行的嵌套限制。这些限制包括以下几点。

- <a>元素中不能包含其他的<a>元素。
- <pre>元素中不能包含<object>、<big>、<img>、<small>、<sub>或<sup>元素。
- <botton>元素中不能包含<input>、<textarea>、<label>、<select>、<botton>、<form>、<iframe>、<fieldset>或<isindex>元素。
- <label>元素中不能包含其他的<label>元素。
- <form>元素中不能包含其他的<form>元素。

3．元素必须要封闭

在 HTML 中，所有的页面元素都要有相应的结束元素。例如，<body>对应的结束元素是</body>。其中独立的元素，例如，
等也必须要结束。方法是在元素的右尖括号前加入一个“/”来结束元素，例如，
就是
结束后的写法。如果元素中还有属性，则“/”出现在所有属性的后面。示例代码如下。

```
<img src="pic.jpg" />
```

4．属性必须加上双引号

HTML 中所有的属性，包括数字值都必须加上双引号。其示例代码如下所示：

```
<table width="400">
```

5．明确所有属性的值

HTML 中规定每一个属性都必须有一个值。没有值的属性也必须用自己的名称作为值。例如，在 HTML 中，checked 属性是可以不取值的，但是在 XHTML 中必须用它自身名称作为值。示例代码如下所示：

```
<input type="checkbox" name="box1" value="abc" checked="checked"   / >
```

6．特殊字符要用编码表示

在 HTML 页面内容中，所有的特殊字符都要用编码表示。比如“&”必须要用“&”的形式。例如，下面的 HTML 代码：

```
<img src="pic.jpg"   src="abc & def">
```

在 XHTML 中必须要写成：

```
<img src="pic.jpg"   src="abc &amp def"   />
```

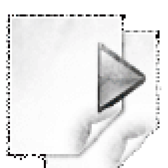
7．推荐使用级联样式表控制外观

在 HTML 中，推荐使用级联样式表控制外观。实现页面的结构和表现相分离，相应的会有部分外观属性不推荐使用，如 algin 属性等(关于级联样式表和属性的知识，将在 4.1.4 节详细介绍)。

8．使用页面注释

HTML 中使用<!--和-->作为页面注释，其示例代码如下所示：

```
<!--这是一个注释 -->
```



说明

在页面中相应的位置使用注释可以使文档结构更加清晰。

9．推荐使用外部链接来调用脚本

HTML 中使用在<!--和-->注释中插入脚本，但是在 XML 浏览器中会被简单地删除，导致脚本或样式的失效。推荐使用外部链接来调用脚本。调用脚本的代码如下所示。

```
<script language="JavaScript1.2" type="text/javascript" src="scripts/menu.js"></script>
```

说明

language 是指所使用的语言的版本。type 是指所使用脚本语言等的种类。src 是指脚本文件所在路径(关于脚本的详细内容，将在 3.8 节详细介绍)。

2.3 文档类型

文档类型(DOCTYPE)的选择，将决定页面中可以使用哪些元素和属性，同时将决定级联样式能否实现。下面详细讲解关于 DOCTYPE 的定义和选择问题。

2.3.1 什么是文档类型

文档类型，又可以写为 DOCTYPE(是 Document Type 的简写)，在页面中用来说明页面所使用的 XHTML 是什么版本。制作 XHTML 页面，一个必不可少的关键组成部分就是 DOCTYPE 声明。只有确定了一个正确的 DOCTYPE，XHTML 里的标识和级联样式才能正常生效。

2.3.2 选择什么样的 DOCTYPE

在 HTML 1.0 中有 3 种 DTD(文档类型定义)声明可以选择：过渡的(Transitional)、严格的(Strict)和框架的(Frameset)。下面分别介绍如下。

1．过渡的

一种要求不很严格的 DTD，允许在页面中使用 HTML 4.01 的标识(符合 XHTML 语法标准)，过渡的 DTD 的写法如下所示。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

2．严格的

一种要求严格的 DTD，不允许使用任何表现层的标识和属性，如
等。严格的 DTD 的写法如下所示。

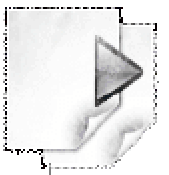
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

3．框架的

一种专门针对框架页面所使用的 DTD，当页面中包含有框架元素时，就要采用这种 DTD。框架的 DTD 的写法如下所示。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
```

使用严格的 DTD 来制作页面当然是最理想的方式，但对于没有深入了解 Web 标准的网页设计者，比较合适的是使用过渡的 DTD。因为这种 DTD 还允许使用表现层的标识、元素和属性。



注意

DOCTYPE 的声明一定要放置在 XHTML 文档的顶部。

在 2001 年 5 月份，W3C 发布了 XHTML1.1 版。其规范和 1.0 版中的严格类型基本相似，其 DTD 的写法如下所示。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
 "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

2.4 什么是名字空间

名字空间的英文是 Namespace，含义就是通过一个网址指向来识别页面上的标签。在 XHTML 中使用的是“xmlns”，也就是 XHTML Namespace 的缩写。用来识别 XHTML 页面上的标签的网址指向是 http://www.w3.org/1999/xhtml。关于名字空间定义的完整写法如下所示。

```
<html xmlns="http://www.w3.org/1999/xhtml">
```

当使用可视化的网页开发工具新建文档时，选择适当的格式的文档类型，DOCTYPE 的声明和名字空间的声明都会自动生成。到目前为止，XHTML 的 4 种文档类型的名字空间都是“http://www.w3.org/1999/xhtml”。

2.5 本章习题

一、选择题

1．用 HTML 标记语言编写一个简单的网页，网页最基本的结构是(　　)。

A. <html> <head>…</head> <frame>…</frame> </html>

B. <html> <title>…</title> <body>…</body> </html>

C. <html> <title>…</title> <frame>…</frame> </html>

D. <html> <head>…</head> <body>…</body> </html>

2．关于 XHTML 的语法规范下面表示错误的是(　　)。

A. XHTML 中对元素的大小写是敏感的。

B. XHTML 中所有的元素都必须封闭。

C. XHTML 中所有元素的属性值都必须用双引号。

D. XHTML 中某些元素的属性可以没有取值。

二、实战练习

1．使用 Dreamweaver 编写不同文档类型的页面。

2．从网上复制内容，并在 Dreamweaver 中查看源文件的写法，熟悉 HTML 的语法。

Chapter 3

页面基本元素

本章主要讲解 HTML 页面基本元素，这些元素是构成页面的基础，它们各有自己相应的作用。掌握这些页面的基本元素是定义 HTML 页面的关键，这些设置决定了页面的文档类型和显示效果，同时也直接影响页面中各种元素能否正常显示。本章的知识点包括<head>元素、<title>元素、<mate>元素、<style>元素和<link>元素等。

本章主要内容有:

◎ <head>元素各个属性的作用和意义。

◎ <style>元素的作用和写法。

◎ <link>元素中各个属性的特点和用法。

3.1 页面基础元素<html>

<html>是页面的基础元素，它通知客户端该文档是 HTML 文档，主要用来定义页面的开始和结束部分，在<html>元素中，包含页面头部和主体部分。语法结构如下所示：

```
<html>整个页面内容</html>
```

在<html>和</html>之间写入用户想要编辑的页面内容就构成了一个简单的页面，下面看一个简单的示例。(在 Dreamweaver 中新建 html 即可)。

例程 3-1 html.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>无标题文档</title>
</head>
<body>
</body>
</html>
```

在这个例程中，<html>元素中定义了 xmlns 属性，用来定义名字空间(关于名字空间的内容，请参照 2.4 节)。

一般来说<html>元素中可以使用的所有属性包括定义名字空间属性 xmlns、文本显示方向属性 dir 和指定语言属性 lang。在后面会逐一进行讲解。

3.1.1 文本显示方向属性 dir

浏览网页时，每一个 HTML 页面的显示都是字体从左向右，浏览器滚条的位置在右方的格式，在 HTML 中字体的位置可以由文本显示方向属性 dir 来控制。dir 属性是用来指定浏览器显示文本的方向，同时也决定浏览器滚条的位置。语法结构如下所示：

```
<html dir="浏览器中文本的方向 ">包含的内容部分</html>
```

dir 属性可以取两个值，即 ltr 和 rtl，它们分别表示浏览器中文本的方向从左向右显示(ltr)和浏览器中文本从右向左显示(rtl)。下面是一个使用 dir 属性的实例。其代码如下所示。

例程 3-2 html-dir.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
< html dir="ltr" html xmlns="http://www.w3.org/1999/xhtml">
<head>
```

```
05 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
06 <title>简单问候</title>
07 </head>
08 <body>
你好呀
09 </body>
10 </html>
```

该实例的 03 行中 dir 属性中使用了 ltr 值，从左向右正常显示。其运行后的显示效果如图 3-1 所示。

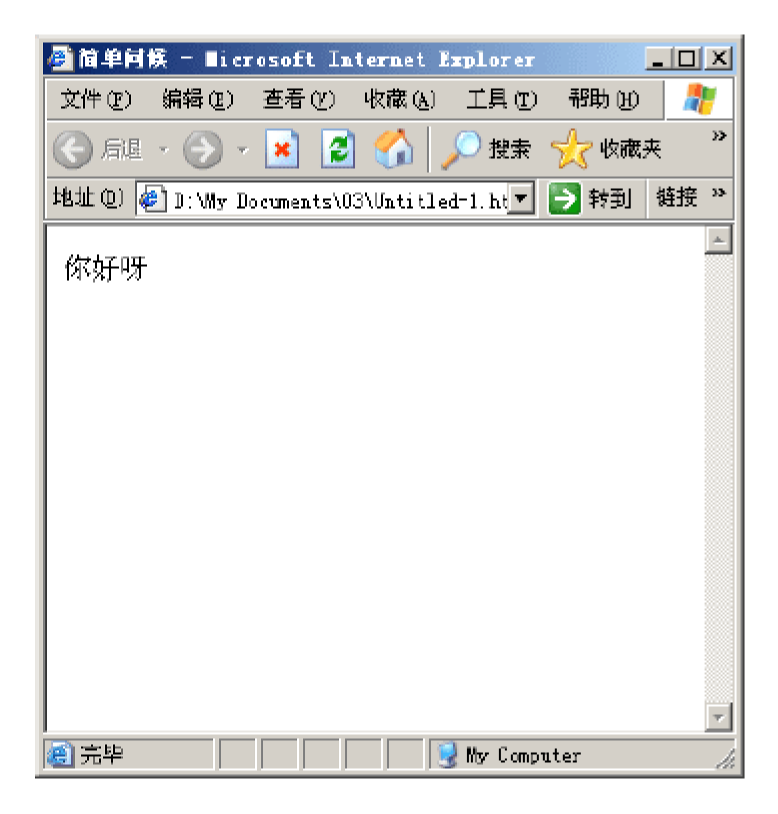


图 3-1　dir 属性取值为 ltr 时的显示效果

上面是正常模式，和通常看到的网页一样的显示。再看下面的例子。

例程 3-3　html-dir.html

```
01 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
02 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
03 <html dir="rtl" xmlns="http://www.w3.org/1999/xhtml">
04 <head>
05 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
06 <title>大家好</title>
07 </head>
08 <body>
初次见面，请多多指教。
09 </body>
10 </html>
```

该实例中，在 03 行中，dir 属性中使用了 rtl 值，从右向左的显示。其运行后的显示效果，如图 3-2 所示。

从图 3-1 和图 3-2 中可以看出，dir 属性只是更改了文本的显示方向，并没有改变文本自身的结构。如图 3-1 所示。文本内容是从左向右的，但是标点中的句号却显示在左侧。同时滚条的位置，从右侧换到了左侧。



图 3-2　dir 属性取值为 rtl 时的显示效果

3.1.2 指定语言属性 lang

lang 属性用来指定文档中所使用的语言。浏览器将会根据指定的语言，更好地显示页面。语法结构如下所示：

```
<html lang="指定的语言 ">包含的内容部分</html>
```

lang 属性的取值可以使用 ISO 标准中的语言代码。常用的语言代码如表 3-1 所示。

表 3-1 常用的语言代码

语 言 名 称	写 法
英语	en
汉语	zh
日语	ja
法语	fr
德语	de
意大利语	it

在<html>元素中加入 lang 属性，使浏览器更好地显示页面，并不会更改页面的文字编码。示例如下。

例程 3-4 html-lang.html

```
01 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
02 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
03 <html lang="fr" xmlns="http://www.w3.org/1999/xhtml">
04 <head>
05 <title>文档标题</title>
06 </head>
07 <body>
   页面内容部分。
08 </body>
09 </html>
```

在 03 行中定义了使用法语作为文档中的使用语言，运行结果如图 3-3 所示。

注意

lang 属性和 3.1.1 节所讲解的 dir 属性，也可以作用在除<html>元素以外的其他元素上，其作用范围为元素所包含的内容部分。



图 3-3 页面的文字编码没有改变

3.2 页面头部元素<head>

在 HTML 语言的头元素中，一般需要包括标题、基底信息、元信息等。HTML 的头部元素是以<head>为开始标记，以</head>为结束标记的。一般情况下，定义在 HTML 语言头部的内容往往不会在网页上直接显示。它用于包含当前文档的相关信息，可以包含<title>元素、<meta>元素等，分别用来定义页面的标题、编码。使用<head>元素可以将基本信息部分和页面主体内容区分开来。语法结构如下所示：

```
<head>包含的其他元素</head>
```

下面是一个使用<head>元素的实例，其代码如下所示。

例程 3-5 head.html

```
01 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
02 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
03 <html xmlns="http://www.w3.org/1999/xhtml">
04 <head>
05 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
06 <title>文档标题</title>
07 </head>
08 <body>
09 </body>
10 </html>
```

该实例中，在 04 到 07 的<head>元素中包含了一个<meta>元素、一个<title>元素，用来定义页面的文字编码和标题。在 HTML 中，允许页面中不使用<head>元素。<head>元素中可以使用的属性也是文本显示方向属性 dir 和指定语言属性 lang。

3.3 页面标题元素<title>

HTML 页面的标题一般是用来说明页面用途的，它显示在浏览器的标题栏中。每个 HTML 页面都应该有标题，在 HTML 文档中，标题信息设置在页面的头部，也就是<head>与</head>之间。标题标记以<title>开始，以</title>结束。语法结构如下所示：

```
<title>…</title>
```

说明

在标记中间的“…”就是标题的内容，它可以帮助用户更好地识别页面。页面的标题有且只有一个。它位于 HTML 文档的头部，即<head>和</head>之间。

下面是一个使用<title>元素的实例，其代码如下所示。

例程 3-6 title.html

```
01  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
02  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
03  <html xmlns="http://www.w3.org/1999/xhtml">
04  <head>
05  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
06  <title>赤壁赋</title>
07  </head>
08  <body>
    赤壁赋苏轼
09  </body>
10  </html>
```

该实例中，在 06 行中<title>元素中定义的页面标题为“赤壁赋”。其运行后，在 IE 浏览器中的显示效果，如图 3-4 所示。

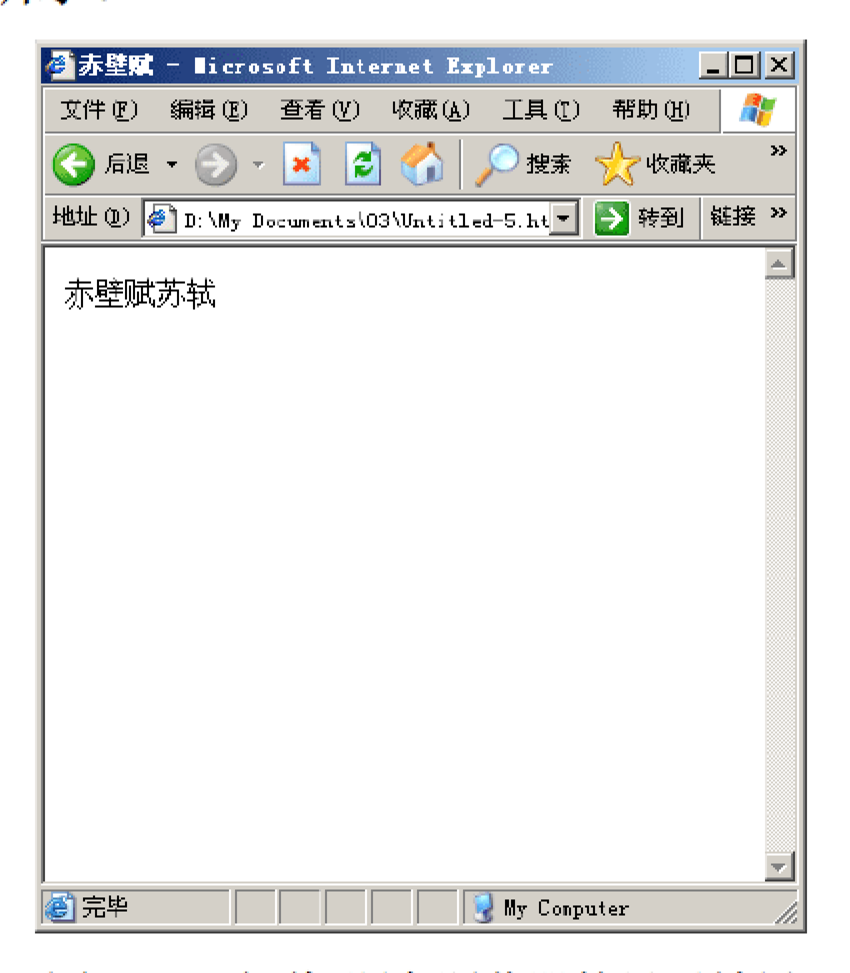


图 3-4　页面标题在浏览器的显示效果

<title>元素中可以使用的所有属性包括文本显示方向属性 dir 和指定语言属性 lang。

3.4 元信息元素<meta>

元信息元素<meta>用来定义页面的附加信息。其中包括页面的作者、版权、关键字等相关

信息。语法结构如下所示：

```
<meta  属性="属性值" />
```

<mate>元素是一个自封闭的元素，通过其中的属性来添加各种附加信息。<mate>元素在不使用任何属性时，对页面没有影响。关于<mate>元素中所使用的属性，将在 3.4.4 节中详细讲解。

3.4.1 元信息元素名称属性 name

name 属性用来指定文档中附加信息的名称。例如，最常用的值“keywords”用来定义文档中的关键字，方便搜索引擎的搜索(关于关键字的内容，请参看 3.4.4 小节)。name 属性的语法结构如下所示：

```
<meta name="信息名称" />
```

因为在<mate>元素中，名称必须对应有相关的值才能生效，所以具体的实例将在讲解完有关值属性后进行演示。

3.4.2 元信息元素的值 content

content 属性用来指定文档中附加信息的值。content 属性是与 name 属性成对出现。用来定义附加信息的具体内容。语法结构如下所示：

```
<mate name="信息名称" content="附加信息的值" />
```

下面是一个使用 content 属性的实例，其代码如下所示。

例程 3-7 meta-content.html

```
01 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
02 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
03 <html lang="en" xmlns="http://www.w3.org/1999/xhtml">
04 <head>
05 <meta name="keywords" content="苏轼" />
06 <title>赤壁赋</title>
07 </head>
08 <body>
09 </body>
10 </html>
11 <body>
12 </body>
13 </html>
```

在实例中，05 行中定义了页面关键字为“苏轼”。因为<meta>元素中所定义的“keywords”信息是用来为搜索引擎定义关键字的，所以对页面显示效果并不产生影响。该实例的显示效果与图 3-4 相同。

3.4.3 元信息元素的附加属性 http-equiv

http-equiv 属性和 name 属性类似，用来指定附加信息的名称。在浏览器加载页面之前，服务器会把 http-equiv 属性定义的相关信息发送给浏览器，便于在浏览器中正确显示页面。语法结构如下所示：

```
<mate http-equiv="信息名称" content="附加信息的值" />
```

和 name 属性类似，http-equiv 属性一般要和 content 属性成对出现。下面是一个使用 http-equiv 属性的实例，其代码如下所示。

例程 3-8 meta-http-equiv.html

```
01 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
02 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
03 <html lang="en" xmlns="http://www.w3.org/1999/xhtml">
04 <head>
05 <meta http-equiv="content-type" content="text/html" />
06 <title>九月</title>
07 </head>
08 <body>
09 </body>
10 </html>
```

在实例中，05 行中使用 http-equiv 属性，提示浏览器将加载一个 HTML 的页面。

3.4.4 设置页面关键字

设置页面关键字是为了向搜索引擎说明这一网页的关键词，从而帮助搜索引擎对该网页进行查找和分类，它可以提高被搜索到的几率，一般可设置不只 1 个关键字，之间用逗号隔开。但是由于很多搜索引擎在检索的时候会限制关键字数量，因此在设置关键字的时候不要过多，注意每一个关键字都要切中要害。语法结构如下所示：

```
<meta name="keywords" content="关键字 1,关键字 2" />
```

上面为设置一个关键字和两个关键字的方法。在 content 属性中设置多个关键字时，每个关键字之间需要用英文“,”分隔开。下面是一个使用关键字的实例，其代码如下所示。

例程 3-9 mate-keywords.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta name="keywords" content="html 入门教程,css 普通教程, java 教程" />
<title>文档标题</title>
</head>
<body>
```

```
   页面内容部分。
09 </body>
10 </html>
```

在上面的实例中，05 行中设定了“html 入门教程”、“css 普通教程”、“java 教程”这 3 个词语作为该页面的关键字。

3.4.5 设置页面主要内容

使用 name 属性和 content 属性，可以设置页面信息的主要内容。设置页面主要内容也是为了便于搜索引擎的查找。页面主要内容可以用来描述网页的主题等，与关键字一样，设置的页面描述也不会在网页中显示出来。其语法结构如下所示：

```
<meta name="description" content="对页面的描述语言">
```

说明

在该语法中，name 为属性名称，这里设置为 description，也就是将元信息属性设置为页面描述，在 content 中定义具体的描述语言。

下面是一个定义主要内容的实例，其代码如下所示。

例程 3-10 mate-description.html

```
01 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
02 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
03 <html xmlns="http://www.w3.org/1999/xhtml">
04 <head>
05 <meta name="description" content="李白简介" />
06 <title>唐代著名诗人</title>
07 </head>
08 <body>
   页面内容部分。
09 </body>
10 </html>
```

在该实例中，05 行中设置了“李白简介”为网页的描述。

3.4.6 定义页面的搜索方式

使用 name 属性和 content 属性，定义页面搜索方式。搜索引擎将根据页面定义的搜索方式，对页面的信息和链接进行搜索。语法结构如下所示：

```
<meta name="robots" content="搜索方式" />
```

在 content 属性中，搜索方式可以有 6 种选择，其具体如下所示。

表 3-2　content 的值及其含义

content 的值	含　义
all	页面将被检索，且页面上的链接可以被查询
none	页面不能被检索，且页面上的链接不可以被查询
index	页面将被检索
follow	页面上的链接可以被查询
noindex	页面不能被检索，但页面上的链接可以被查询
nofollow	页面不能被检索，页面上的链接可以被查询

下面是一个使用搜索方式的实例，其代码如下所示。

例程 3-11　mate-robots.html

```
01  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
02  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
03  <html xmlns="http://www.w3.org/1999/xhtml">
04  <head>
05  <meta name="robots" content=" noindex " />
06  <title>全唐诗</title>
07  </head>
08  <body>
    页面内容部分。
09  </body>
10  </html>
```

这个示例中 05 行中设定了页面不能被检索，但页面上的链接可以被查询。

3.4.7　定义页面的跳转

在浏览网页时经常会看到一些欢迎信息的界面，在经过一段时间后，这一页面会自动转到其他页中，这就是网页的跳转。使用 HTML 中的 HTTP 代码就可以很轻松地实现这一功能。语法结构如下所示：

```
<meta http-equiv="refresh" content="跳转时间;url=链接地址">
```

说明

在该语法中，refresh 表示网页的刷新，而在 content 中则设定的是刷新的时间和刷新后的地址，时间和链接地址之间用分号相隔。默认情况下，跳转时间是以秒为单位的。

当链接地址为一个新的网页地址时，就会在设定的时间跳转到这个新的网址。下面是一个使用页面的跳转的实例，其代码如下所示。

例程 3-12 mate-refresh-url.html

```
01  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
02  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
03  <html xmlns="http://www.w3.org/1999/xhtml">
04  <head>
05  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
06  <meta http-equiv="refresh" content="5;url=http://www.hao123.com">
07  <title>全唐诗</title>
08  </head>
09  <body>
您好，页面将在 5 秒后跳转至 http://www.hao123.com。
10  </body>
11  </html>
```

上面的代码在 06 行中设置了页面跳转。运行上面代码的时候显示如图 3-5 所示。在 5 秒之后，网页自动跳转到 www.hao123.com 网站，如图 3-6 所示。



图 3-5 跳转前的页面



图 3-6 跳转后的页面

3.4.8 定义页面的作者信息

在 html 中使用 name 属性和 content 属性来定义页面的作者信息。语法结构如下所示：

```
<meta name=" author " content="作者" />
```

下面是一个定义作者信息的实例，其代码如下所示。

例程 3-13 mate-author.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<meta name=" author " content="张三" />
<title>作者简介</title>
```

```
08  </head>
09  <body>
    全唐诗讲解的作者张三。
10  </body>
11  </html>
```

上面的实例中，06 行中将作者的姓名“张三”添加到了网页的源代码中。

3.4.9 定义页面的版权信息

name 属性和 content 属性也可以来定义页面所使用的版权。语法结构如下所示：

```
<meta name=" Copyright" content="版权" />
```

下面是一个使用版权信息的实例，其代码如下所示。

例程 3-14 mate-copyright.html

```
01  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
02  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
03  <html xmlns="http://www.w3.org/1999/xhtml">
04  <head>
05  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
06  <meta name=" Copyright" content="版权" />
07  <title>版权所有</title>
08  </head>
09  <body>
    此页面页面版权为张三所有。
11  </body>
12  </html>
```

运行代码后，06 行中提示此页面的版权为张三所有。

3.4.10 定义页面的刷新时间

人们在游览网页时经常会遇到一些定时刷新的页面。这些页面采用动态更新的方式显示，所以每间隔一定的时间就会自动刷新页面，可以使浏览者方便看到最新添加的信息。在 HTML 中可以用 http_equiv 属性和 content 属性定义页面刷新时间。其语法结构如下所示。

```
<meta http-equiv=" refresh " content="时间" />
```

刷新页面所使用的时间单位是秒。下面是一个使用刷新时间的实例，其代码如下所示。

例程 3-15 mate-refresh.html

```
01  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
02  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
03  <html xmlns="http://www.w3.org/1999/xhtml">
04  <head>
05  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
```

```
06  <meta http-equiv=" refresh " content="6" />
07  <title>千家诗</title>
08  </head>
09  <body>
    页面内容部分。
10  </body>
11  </html>
```

在上面的代码中，06 行定义该页面 6 秒刷新一次。

3.5 基本设置元素<base>

在 HTML 中基本设置元素<base>用来定义相对路径的根目录。使用<base>元素，可以方便地定义页面中的超级链接。语法结构如下所示：

```
<base  属性="属性值" />
```

<base>元素在不使用任何属性时，对页面没有影响。关于<base>元素中所使用的属性，将在第 8 章中详细讲解。<base>元素中可以使用的属性有链接路径属性 href 和链接窗口属性 target。

href 属性用来指定文档中相对链接的根目录。文档中的所有链接(包括图片、音频等内容)都按照 href 属性所指定的根目录显示。语法结构如下所示：

```
<base href="指定路径 ">包含的内容部分</base >
```

href 属性的取值为 URL 值(关于 URL 值的详细内容，请参看第 8 章)。它可以使用绝对路径，也可以指向某个文件夹。下面是一个 href 属性取值为绝对路径的实例。其代码如下所示。

例程 3-16　base-href.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<base href="http://www.baidu.com" />
<title>页链接路径</title>
</head>
<body>
含有链接的图片：<img src="http://www.baidu.com/img/baidu_logo.gif" alt="pic" />
</body>
</html>
```

说明

09 行代码中，含有链接的元素为<img>元素(关于<img>元素的详细内容，请参看第 6 章)。其中 src 属性的值，就是链接的路径。

该实例中，使用<base>元素定义页面所有链接的根目录为“http://www. baidu.com”。其运行后的显示效果如图 3-7 所示。

在页面<img>元素中，使用相对路径，调用了“www.baidu.com”上的一个图片。使用的相对路径值为“http://www.baidu.com/img/baidu_logo.gif”。



图 3-7　定义<base>元素链接属性后的页面

3.6 创建样式元素<style>

创建样式元素<style>用来创建本页面中所使用的样式(关于样式的详细内容，请参见本书 11.3 节)。使用<style>元素创建的样式内容只能够被当前页面使用。语法结构如下所示。

```
<style>级联样式</style>
```

下面是一个使用<style>元素的实例，其代码如下所示。

例程 3-17　style.html

```
01  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
02  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
03  <html xmlns="http://www.w3.org/1999/xhtml">
04  <head>
05  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
06  <title>样式说明</title>
07  <style>
08  body{
    color:#00FF33;
    font-size:18px;
    font-weight:bold;}
09  </style>
10  </head>
11  <body>
请注意这个页面文本的显示效果。
12  </body>
13  </html>
```

在以上代码中，在 07 到 09 之间的<style>元素通过级联样式定义了页面主体中文本的显示

方式，并定义字体的颜色为绿色，字体大小为 18 像素，字体显示为加粗。其运行后，显示效果如图 3-8 所示。

3.6.1　类型属性 type

type 属性，用来指定<style>元素中所包含内容的类型。一般情况下，不指明 type 属性时，浏览器是可以辨别内容的类型的。但为了防止和 js 脚本等混淆，最好定义 type 属性。语法结构如下所示：

```
<style type="类型"></style>
```

在<style>元素中，类型属性的值为“text/css”。下面是一个使用类型属性的实例，其代码如下所示。

例程 3-18　style-type.html

```
01  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
02  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
03  <html xmlns="http://www.w3.org/1999/xhtml">
04  <head>
05  <title> type 属性说明</title>
06  <style type="text/css">
07  body{
    color:# CC0000;
    font-size:26px;
    font-weight:bold;}
08  </style>
09  </head>
10  <body>
不识庐山真面目，只缘身在此山中。
11  </body>
12  </html>
```

在以上代码中，06 到 08 行通过<style>元素定义字体颜色为紫色，字体为 26 号，字体显示为加粗。其运行后显示效果如图 3-9 所示。其与不指明类型属性的页面效果显示基本相同。



图 3-8　使用<style>元素后的显示效果



图 3-9　页面显示效果

3.6.2 类型属性 media

media 属性用来指定<style>元素中所包含内容作用的媒体。因为一般情况下，制作的网页可能在不同的浏览设备中使用，指明使用的媒体将有助于页面的显示。其语法结构如下所示。

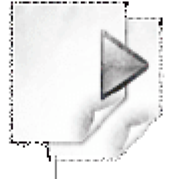
```
<style media="媒体类型"></style>
```

在 media 属性中，可以使用多种媒体，具体的取值如表 3-3 所示。

表 3-3　media 属性的取值及含义

代码属性值	含　义
screen	电脑的显示器
tv	电视
print	打印机
aural	音频设备
tty	文本
all	所有设备

当指定多个媒体时，每个媒体值之间，用英文的“,”分隔。



注意

当指定了使用媒体后，其他媒体中将不能应用现有样式。

下面是一个使用 media 属性的实例，其代码如下所示。

例程 3-19 style-media.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>使用 media 属性</title>
<style media="sreen,tv">
body{
color:# CC0000;
font-size:26px;
font-weight:bold;}
</style>
</head>
<body>
请注意页面的显示效果。
</body>
```

```
13  </html>
```

07 行用<style media="sreen,tv ">定义使用的多媒体工具为电脑显示器和电视机。

3.7 链接元素<link>

链接元素<link>用来指定文档与其他文档之间的关系。使用<link>元素，可以用来调用其他文档中的内容，如级联样式表等。其语法结构如下所示。

```
<link 属性="属性值" />
```

下面是一个使用<link>元素的实例，其代码如下所示。

例程 3-20 link.html

```
01  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
02  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
03  <html xmlns="http://www.w3.org/1999/xhtml">
04  <head>
05  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
06  <title>全唐诗</title>
07  <link href=" main.css" rel="stylesheet" type="text/css" />
08  </head>
09  <body>
注意页面文本内容部分的显示效果。
10  </body>
11  </html>
```

07 行中代码在标题中链接了一个 main.css 的样式(具体应用请参考第 12 章)。<link>元素中可以使用的所有属性如表 3-4 所示。

表 3-4 <link>元素的所有属性

属性名称	写法
文本显示方向属性	dir
指定语言属性	lang
类型属性	type
媒体选择属性	media
标题属性	title
字符集名称属性	charset
指定链接路径属性	href
源文档与目标文档关系属性	rel
目标文档与源文档关系属性	rev

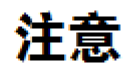
(续表)

属 性 名 称	写 法
链接窗口属性	target
标记属性	id
类属性	class
定义级联样式属性	style

3.7.1 指定链接路径属性 href

href 属性用来指定<link>元素中链接文档的路径。其语法结构如下所示。

```
<link href="指定的路径" />
```



注意

在<link>元素中，href 属性是必须的。其取值可以是绝对路径也可以是相对路径(关于路径的知识，请参见第 8 章)。

下面是一个使用 href 属性的实例，其代码如下所示。

例程 3-21 link-href.html

```
01  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
02  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
03  <html xmlns="http://www.w3.org/1999/xhtml">
04  <head>
05  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
06  <title>href 属性说明</title>
07  <link href="style/main.css" />
08  </head>
09  <body>
这是页面文本内容部分，注意文本的显示效果。
10  </body>
11  </html>
```

07 行代码在标题中链接了一个 main.css 的样式(具体应用参考第 12 章)。

3.7.2 链接的类型属性 type

type 属性用来指定<link>元素中链接文档的类型。其语法结构如下所示。

```
<link type="文档类型" />
```

下面是一个使用 type 属性的实例，其代码如下所示。

例程 3-22 link-type.html

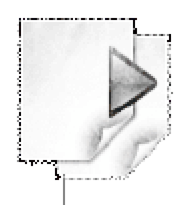
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
```

```
02  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
03  <html xmlns="http://www.w3.org/1999/xhtml">
04  <head>
05  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
06  <title>文档标题</title>
07  <link href=" main.css"   type="text/css" />
08  </head>
09  <body>
这是页面文本内容部分，注意文本的显示效果。
10  </body>
11  </html>
```

07 行代码在标题中链接了一个类型为“text/css”的 css 样式(具体应用参考第 12 章)。

3.7.3 源文档与目标文档关系属性 rel、rev

rel 属性用来指定源文档到链接文档之间的关系。rev 属性用来指定链接文档到源文档之间的关系。其语法结构如下所示。



注意

rel 属性并不是 XHTML 标准中定义的属性，但是在定义链接时经常用到。

```
<link rel="文档关系" />
```

rel 属性和 rev 属性可以同时使用，它们的使用方法也相同。它们的部分取值和含义如表 3-5 所示。

表 3-5 rel 属性和 rev 属性的部分取值和含义

属 性 值	代表的含义
next	链接到下一个文档
prev	链接到前一个文档
head	链接到集合中的顶级文档
toc	链接到集合的目录
parent	链接到源上面的文档
child	链接到源下面的文档
index	链接到此文档的索引
glossary	链接到此文档的术语表
stylesheet	链接到样式片段

下面是一个使用 rel 属性的实例，其代码如下所示。

例程 3-23 link-rel.html

```
01    <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
02    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
03    <html xmlns="http://www.w3.org/1999/xhtml">
04    <head>
05    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
06    <title>文档标题</title>
07    <link href="style/main.css" rel="stylesheet" />
08    </head>
09    <body>
这是页面文本内容部分，注意文本的显示效果。
10    </body>
11    </html>
```

在该实例中，07 行中使用 rel 属性定义文档之间的关系为样式片段。

3.7.4 链接样式文件

通过<link>元素及其相应的属性，实现调用级联样式表文件的目的。其语法结构如下所示。

```
<link href="样式路径 " rel="文档关系" type="文件类型" />
```

下面是一个定义链接样式文件的实例。其代码如下所示。

例程 3-24 link-style.html

```
01    <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
02    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
03    <html xmlns="http://www.w3.org/1999/xhtml">
04    <head>
05    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
06    <title>link 属性</title>
07    <link href="main.css" rel="stylesheet" type="text/css" />
08    </head>
09    <body>
这是页面文本内容部分，注意文本的显示效果。
10 </body>
11    </html>
```

在以上代码中，07 行中使用<link>元素调用了一个名称为 main.css 的样式文件。该文件中具体定义的样式内容如下所示。

```
body
{
  width:200px;
  height:120px;
  background:#999999
  }
```

说明

该样式中，定义<body>元素的背景色为蓝绿色，文本颜色为黑色，字体大小为24像素，字体样式为加粗。

代码运行后，其显示效果如图3-10所示。

在以上的代码中，<link>元素的rel属性取值为stylesheet，表明了将链接的文件为样式片段文件。如果取消rel属性，则页面的显示效果如图3-11所示。

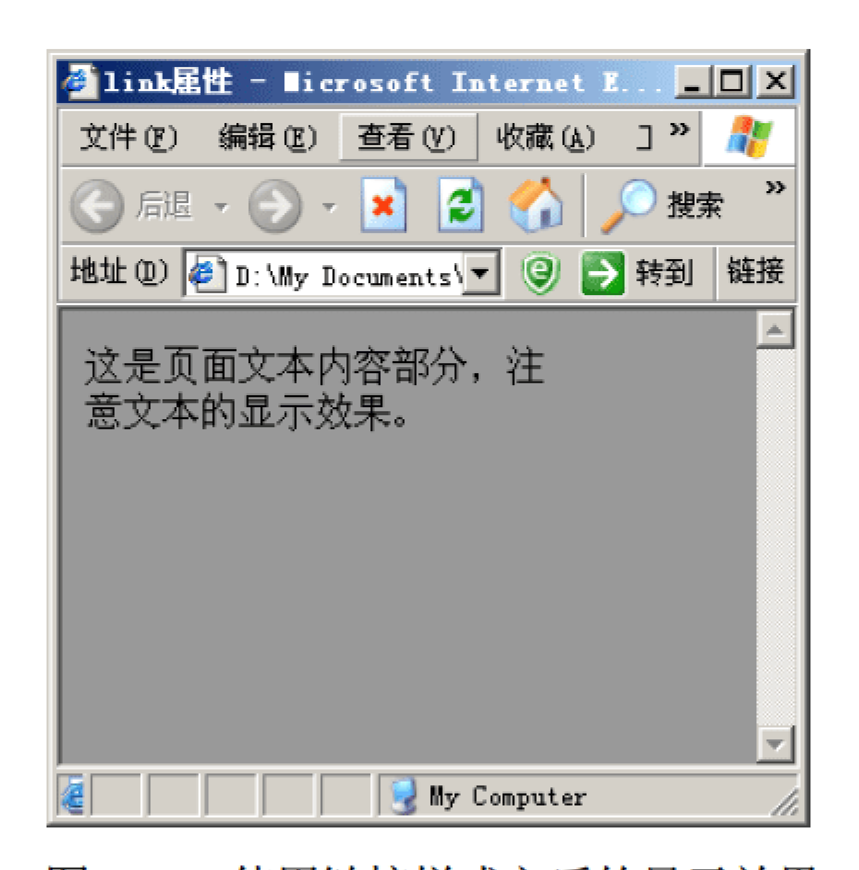



图3-10 使用链接样式之后的显示效果

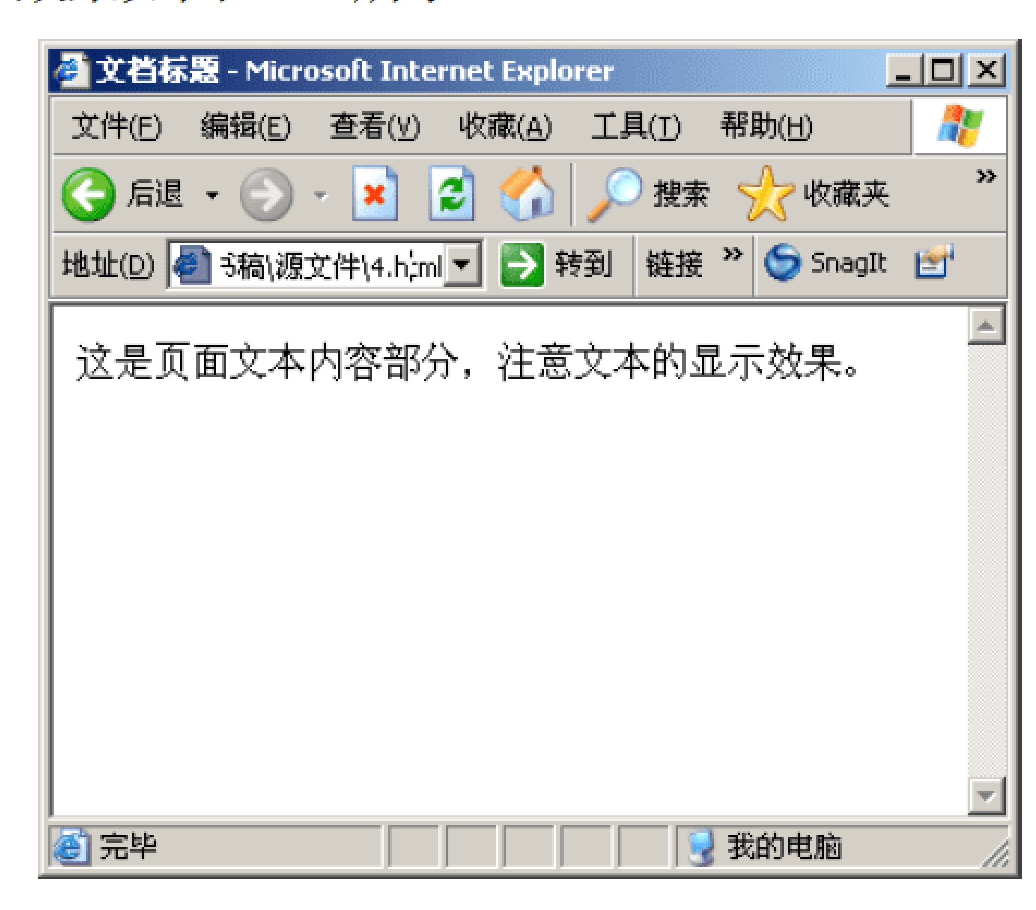


图3-11 取消rel属性后的显示效果

从以上实例可以看出，虽然rel属性不是XHTML标准中的属性，但是对能否正常调用链接文件有很大的影响。

3.7.5 制作收藏夹图标

通过<link>元素及其相应的属性，制作收藏夹中的图标。其语法结构如下所示。

```
<link rel="Shortcut Icon" href="图标路径 " type="image/x-icon" />
```

在制作收藏夹图标时，<link>元素中，rel属性的值为"shortcut icon"，类型为"mage/x-icon"。同时图标要使用相应的icon格式图标。下面是一个制作收藏夹图标的实例。其代码如下所示。

例程 3-25 link-shortcut.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>文档标题</title>
<link rel="Shortcut Icon" href="http://www.w3pop.com/favicon.ico" type="image/x-icon" />
</head>
<body>
```

```
这是页面文本内容部分。
10   </body>
11   </html>
```

代码中 07 行定义了收藏夹。代码运行后，将页面添加到收藏夹中，其在 IE 浏览器中的显示效果，如图 3-12 所示。

从图 3-12 中可以看出，此时页面在收藏夹中显示出特有的图标。

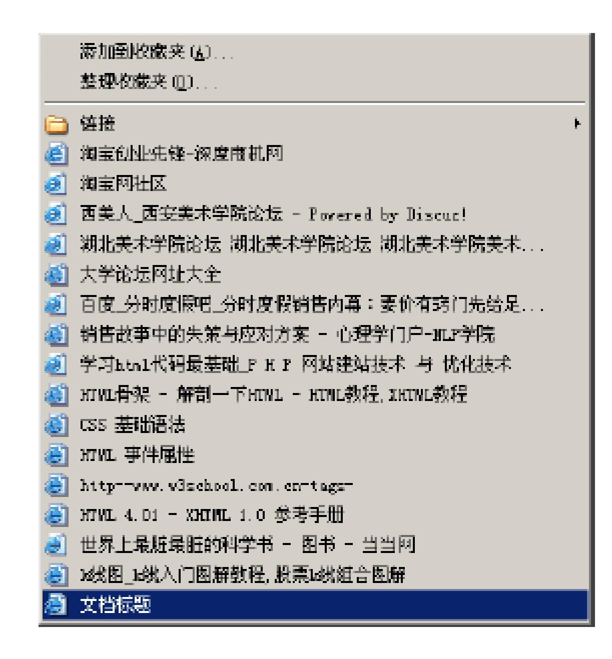


图 3-12　制作后的收藏夹显示效果

3.8 脚本元素<script>

脚本元素<script>，用来在页面中定义一个可执行的脚本，完成相应的行为，如鼠标点击事件等。语法结构如下所示。

```
<script>脚本内容</script>
```

<script>元素，不但可以使用在文档头部，也可以用于文档的主体部分。下面是一个在文档主体中使用<script >元素的实例，其代码如下所示。

例程 3-26　script.html

```
01   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
02   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
03   <html xmlns="http://www.w3.org/1999/xhtml">
04   <head>
05   <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
06   <title>脚本讲解</title>
07   </head>
08   <body>
接下来页面出现的<script>
<!--
09   document.write("<strong>脚本内容</strong>");
 -->
10   </script>
11   </body>
12   </html>
```

在该实例中，使用脚本实现了将内容输出到浏览器的效果。其中 09 行中<strong>元素用来定义文本加粗(关于<strong>元素的详细内容，请参看第 4 章)。<script>元素中所使用属性的含义，将在 3.8.1 节和 3.8.2 节中具体介绍。代码运行后，其显示效果如图 3-13 所示。

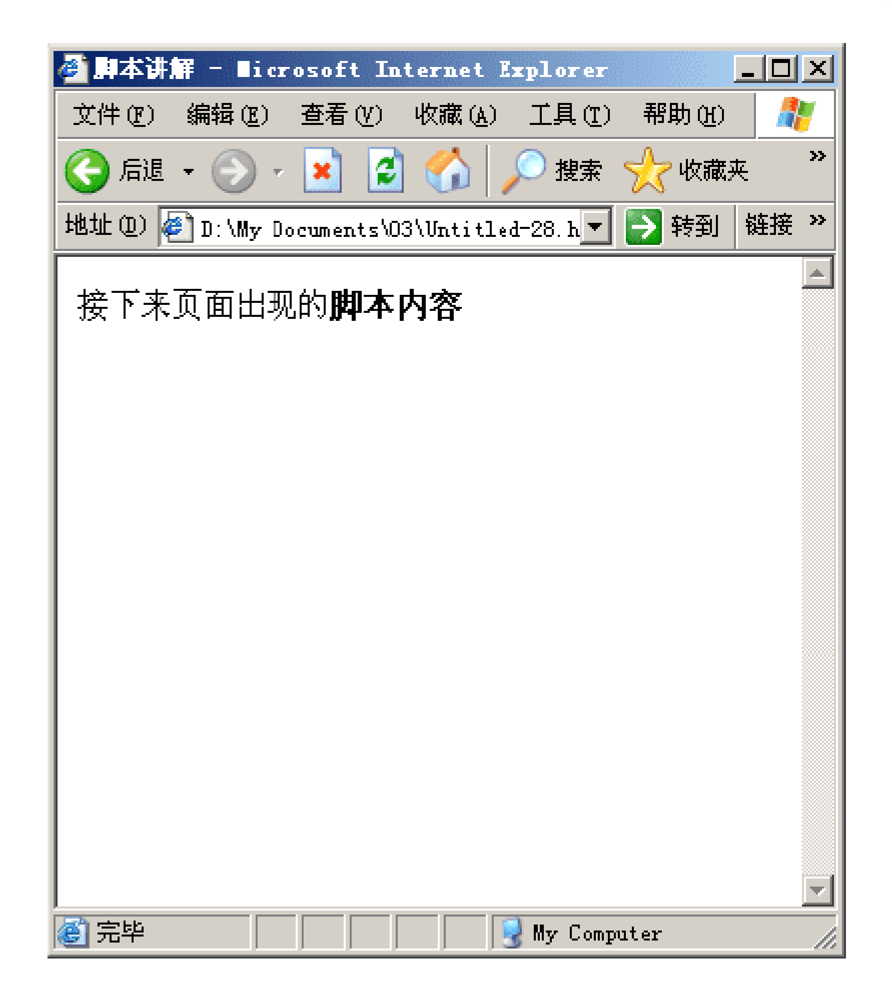


图 3-13　使用<script>元素的显示效果

3.8.1　脚本的语言属性 language

language 属性的作用是：定义脚本中使用的语言。其语法结构如下所示。

```
<script language="脚本语言">脚本内容</script>
```

在制作页面时，一般常用的脚本语言有两种。一种是 JavaScript，另一种是 VBScript。下面是使用 language 属性的实例，其代码如下所示。

例程 3-27　script-language.html

```
01  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
02  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
03  <html xmlns="http://www.w3.org/1999/xhtml">
04  <head>
05  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
06  <title>脚本语言属性</title>
07  </head>
08  <body>
此时页面显示的是：
09  <script language="VBScript">
<!--
document.write("<strong>脚本内容</strong>");
 -->
10  </script>
11  </body>
12  </html>
```

该实例中的脚本和例程 3-26 中基本相同，只是在 09 行的<script>元素中加入了 language 属性。代码运行后的显示效果和例程 3-26 基本相同。

3.8.2　脚本的类型属性 type

type 属性的作用是定义脚本中使用的语言类型。其语法结构如下所示。

```
<script type="脚本语言类型">脚本内容</script>
```

type 属性的作用和 language 属性基本相同，不过在属性值的写法上有所区别。其中属性值的具体写法如下所示。

例程 3-28　script-type.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>脚本类型讲解</title>
</head>
<body>
接下来显示的是：
<script type="text/vbscrip">
<!--
 document.write("<strong>脚本内容</strong>");
 -->
</script>
</body>
</html>
```

代码运行后的显示效果和例程 3-26 基本相同。

3.8.3　推迟执行属性 defer

defer 属性使脚本中不显示的内容推迟执行。目的是加快页面的显示速度。其语法结构如下所示。

```
<script defer=defer>脚本内容</script>
```

defer 属性是没有值的，按照 HTML 的语法规则，要使用自身名称作为值。因为 defer 属性用于非显示的脚本内容，所以本书在此就不做实例演示了。

3.8.4　脚本的链接属性 src

src 属性的作用是链接外部的脚本文件。其语法结构如下所示。

```
<script src="脚本文件的路径"> </script>
```

在 XHTML 中，推荐使用外部链接的脚本。下面是使用 src 属性的实例，其代码如下所示。

例程 3-29　script-src.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
```

```
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>脚本类型</title>
</head>
<body>
接下来显示的是：
<script language="JavaScript" type="text/javascript" src="content.js">
<!--
 document.write("<strong>脚本内容</strong>");
 -->
</script>
</body>
</html>
```

在“content.js”中，使用的脚本如下所示。

```
<!--
 document.write("<strong>脚本内容</strong>");
 -->
```

代码运行后的显示效果和例程 3-26 中基本相同。

3.9 页面主体元素<body>

页面主体元素<body>用来包含页面所要显示的内容。包括文本元素、图像元素、表单元素等各种页面元素。同时可以设置整个页面的背景、边界等相关属性。语法结构如下所示。

```
<body>页面主体内容</body>
```

下面是一个使用<body>元素的实例，其代码如下所示。

例程 3-30 body.html

```
01 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
02 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
03 <html xmlns="http://www.w3.org/1999/xhtml">
04 <head>
05 <meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
06 <title>文档标题</title>
07 </head>
08 <body>
页面主体中的内容
09 </body>
10 </html>
```

在不定义任何属性时，08 和 09 行中<body>元素中的内容会和浏览器的边线保持一定的距离(在不同的浏览器中这个距离是不同的)。下面是该实例运行后，在 IE 浏览器中的显示效果，

如图 3-14 所示。

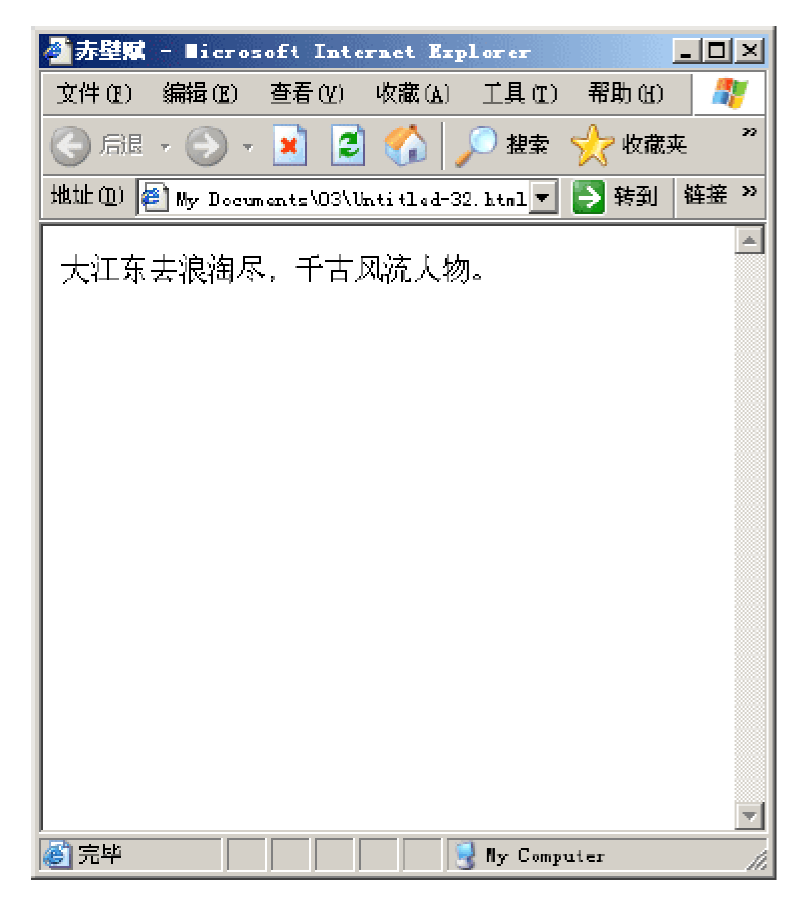


图 3-14　<body>元素在 IE 浏览器中的显示效果

<body>元素中可以使用的所有属性如表 3-6 所示。

表 3-6　<body>元素的所有属性

属 性 名 称	写　法
文本显示方向属性	dir
指定语言属性	lang
背景图片属性	background
背景颜色属性	bgcolor
标题属性	title
文本属性	charset
指定链接路径属性	text
链接属性	link
已链接属性	vlink
激活链接属性	alink
标记属性	id
类属性	class
定义级联样式属性	style
左边界属性	leftmargin
上边界属性	topmargin

<body>元素的背景和文本属性是制作页面时常用的属性。内容包括 bgcolor 属性、

background 属性、bgproperties 属性和 text 属性，下面进行详细讲解。

3.9.1　主体元素的背景属性 bgcolor

bgcolor 属性用来指定页面所使用的背景颜色。其语法结构如下所示。

```
<body bgcolor="颜色值">包含的内容部分</body>
```

bgcolor 属性的取值，可以使用颜色名称，也可以使用 16 进制的颜色代码值。16 进制的颜色代码的含义是，使用 6 位的数字(如“#ff9900”)来定义颜色值。其中，每连续两个数字代表一种原色值。原色按照“红、绿、蓝”的顺序排列。数值越大，代表颜色所占的比例越大。

另外，在使用颜色名称时，并不是所有的颜色名称都能够正常显示。bgcolor 属性的具体取值，如表 3-7 所示。

表 3-7　bgcolor 属性的取值及含义

属 性 值	含　义	属 性 值	含　义
red	红色	purple	紫
yellow	黄	gray	灰
blue	蓝	lime	浅绿
silver	银	maroon	褐
teal	深青	aqua	水绿
white	白	black	黑
navy	深蓝	fuchsia	紫红
olive	橄榄	green	绿
#ff9900	十六进制颜色		

下面是一个使用 bgcolor 属性的实例，其实例代码如下所示。

例程 3-31　body-bgcolor-name.html

```
01 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
02 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
03 <html xmlns="http://www.w3.org/1999/xhtml">
04 <head>
05 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
06 <title>赤壁赋</title>
07 </head>
08 <body bgcolor="yellow">
大江东去浪淘尽，千古风流人物。
09 </body>
10 </html>
```

在实例中，08 行中使用的颜色值为 yellow。其运行后的显示效果，如图 3-15 所示。使用

颜色名称的颜色值来定义页面背景，可以选择的背景是很少的。如果想制作出更加精彩的背景颜色，可以使用 16 进制的颜色值。下面是一个使用 16 进制的颜色值的实例，其代码如下所示。

例程 3-32　body-bgcolor-16.html

```
01  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
02  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
03  <html xmlns="http://www.w3.org/1999/xhtml">
04  <head>
05  <meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
06  <title>赤壁赋</title>
07  </head>
08  <body bgcolor="#FFFF00">
大江东去浪淘尽，千古风流人物。
09  </body>
10  </html>
```

在实例中，08 行中使用的颜色值为“#FFFF00”，也是黄色的。其运行后的显示效果如图 3-16 所示。



图 3-15　bgcolor 属性中使用颜色名称的表现效果

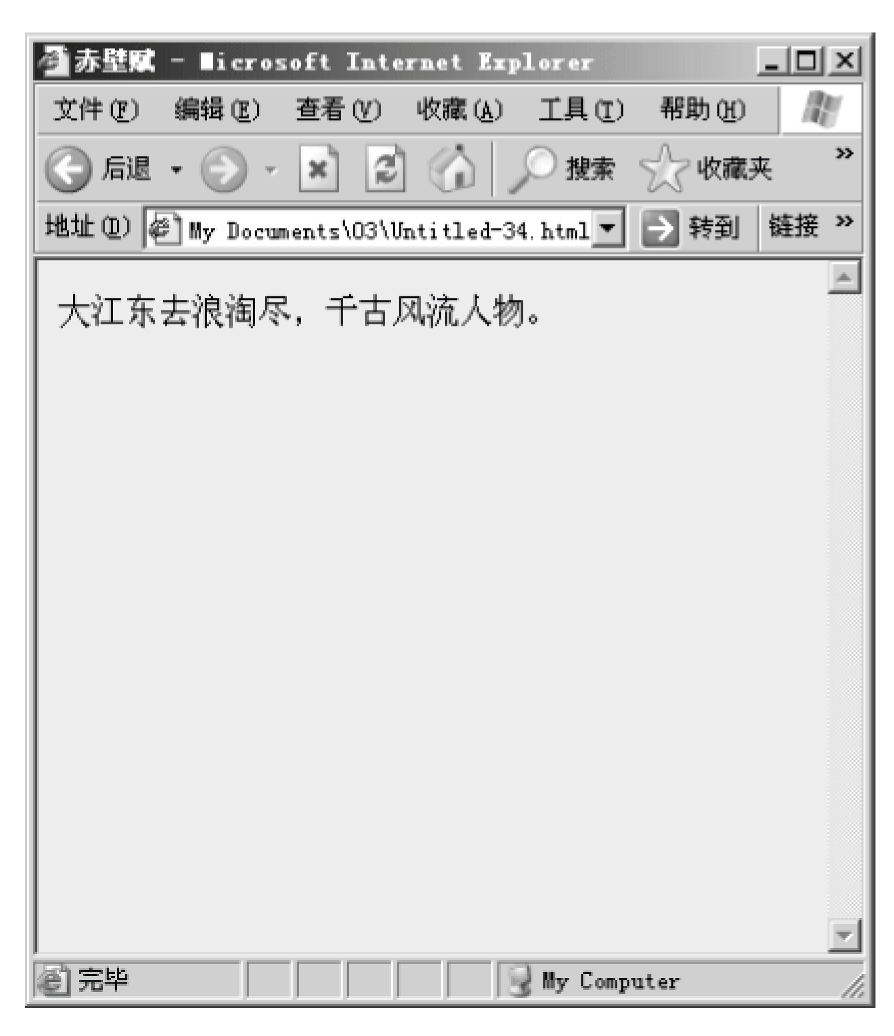


图 3-16　bgcolor 属性中使用 16 进制的颜色的表现效果

3.9.2　主体元素的背景图片属性 background

background 属性用来指定页面所使用的背景图片。指定的背景图片，将按照从上到下，从左到右的方式重复显示。语法结构如下所示。

```
<body background="图片的路径">包含的内容部分</body>
```

background 属性的取值为 URL 值。它可以使用绝对路径，也可以使用相对路径(关于 URL 值的详细内容，请参看第 8 章)。下面是一个使用 background 属性的实例。其代码如下所示。

例程 3-33　body-background.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
```

```
02 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
03 <html xmlns="http://www.w3.org/1999/xhtml">
04 <head>
05 <meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
06 <title>文档标题</title>
07 </head>
08 <body background="http://www.baidu.com/img/baidu_logo.gif">
页面主体中的内容
09 </body>
10 </html>
```

该实例中，08 行中使用 background 属性，定义的页面所使用的背景图片为 baidu 站点的 logo。其运行后的显示效果如图 3-17 所示。



图 3-17　定义 background 属性后的显示效果

在 background 属性中定义的背景图片，不能控制其重复、位置等属性。关于背景图片的精确控制，请参看本书第 8 章内容。

3.9.3　主体元素的背景图片滚动属性 bgproperties

bgproperties 属性用来指定背景图片能否滚动。背景图片滚动的含义是，当拖动浏览器滚条时，页面背景可以随滚条一起改变位置。语法结构如下所示。

```
<body background="图片的路径" bgproperties="fixed">包含的内容部分</body>
```

因为 bgproperties 属性是用来控制背景图片的，所以一定要和 background 属性一起使用。bgproperties 属性只有一个值 fixed，含义是背景图片固定。下面是使用 bgproperties 属性的实例。其代码如下所示。

例程 3-34　body-bgproperties.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
```

```
05   <meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
06   <title>文档标题</title>
07   </head>
08   <body background=" http://www.baidu.com/img/baidu_logo.gif " bgproperties="fixed">
09   <table height="400">
     <tr>
     <td>页面主体中的内容</td>
</tr>
10   </table>
11   </body>
12   </html>
```

该实例中 09 到 10 行使用了<table>元素，用来产生滚条(关于<table>元素的详细内容，请参看第 7 章)。其运行后的显示效果如图 3-18 所示。当拖动滚条后，其显示效果如图 3-19 所示(注意滚条所在的位置)。





图 3-18　使用 bgproperties 属性的表现效果



图 3-19　拖动滚条后的表现效果

从图 3-19 可以看出，当使用了 bgproperties 属性后，拖动滚条时，背景保持不动。如果取消 bgproperties 属性，当拖动滚条到相同位置时，页面的显示效果如图 3-20 所示。



图 3-20　取消 bgproperties 属性后的显示效果

3.9.4 主体元素的文本属性 text

text 属性的作用是：用来指定页面中普通文本(没有链接的文本)的颜色。其语法结构如下所示。

```
<body text="颜色值">包含的内容部分</body>
```

下面是一个使用 text 属性的实例，其代码如下所示。

例程 3-35 body-text.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>赤壁怀古</title>
</head>
<body bgcolor="#333333" text="#ffffff">
大江东去，浪淘尽，千古风流人物。 故垒西边，人道是、三国周郎赤壁。 乱石穿空，惊涛拍岸，卷起千堆雪。 江山如画，一时多少豪杰！
        遥想公瑾当年，小乔初嫁了，雄姿英发。 羽扇纶巾，谈笑间、樯橹灰飞烟灭。 故国神游，多情应笑我，早生华发。 人生如梦，一樽还酹江月。
</body>
</html>
```

在实例中，定义了背景色为黑色，文字的颜色为白色。其运行后的显示效果，如图 3-21 所示。

<body>元素的边界属性用来定义页面与浏览器窗口之间的距离。边界属性包括两个属性，leftmargin 属性和 topmargin 属性。

> **注意**
>
> 这两个属性都是 IE 浏览器所专有的。如果想在其他浏览器中达到一致的显示效果，建议使用级联样式表实现。

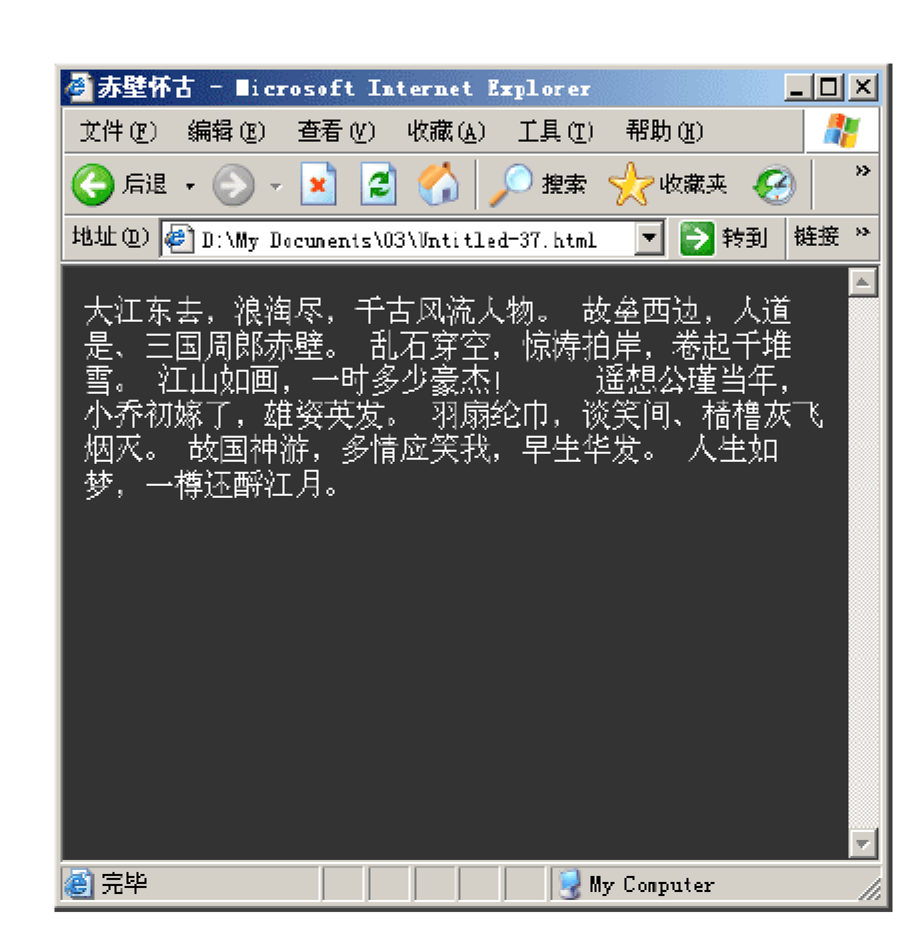


图 3-21 使用 text 属性后的显示效果

3.9.5 IE 浏览器中的左边界属性 leftmargin

leftmargin 属性用来指定页面和浏览器左边界之间的距离。语法结构如下所示。

```
<body leftmargin="数字值">包含的内容部分</body>
```

在 leftmargin 属性中，取值用数字表示，其实际单位是像素。示例代码如下所示。

例程 3-36 body-leftmargin.html

```
01 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
02 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
03 <html xmlns="http://www.w3.org/1999/xhtml">
03 <head>
04 <meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
05 <title>文档标题</title>
06 </head>
07 <body leftmargin="0">
页面主体中的内容
08 </body>
10 </html>
```

在实例中，07 行定义了页面与浏览器左侧的距离为 0。运行后，在 IE 浏览器中的显示效果如图 3-22 所示。该实例在 Firefox 浏览器中的显示效果如图 3-23 所示。

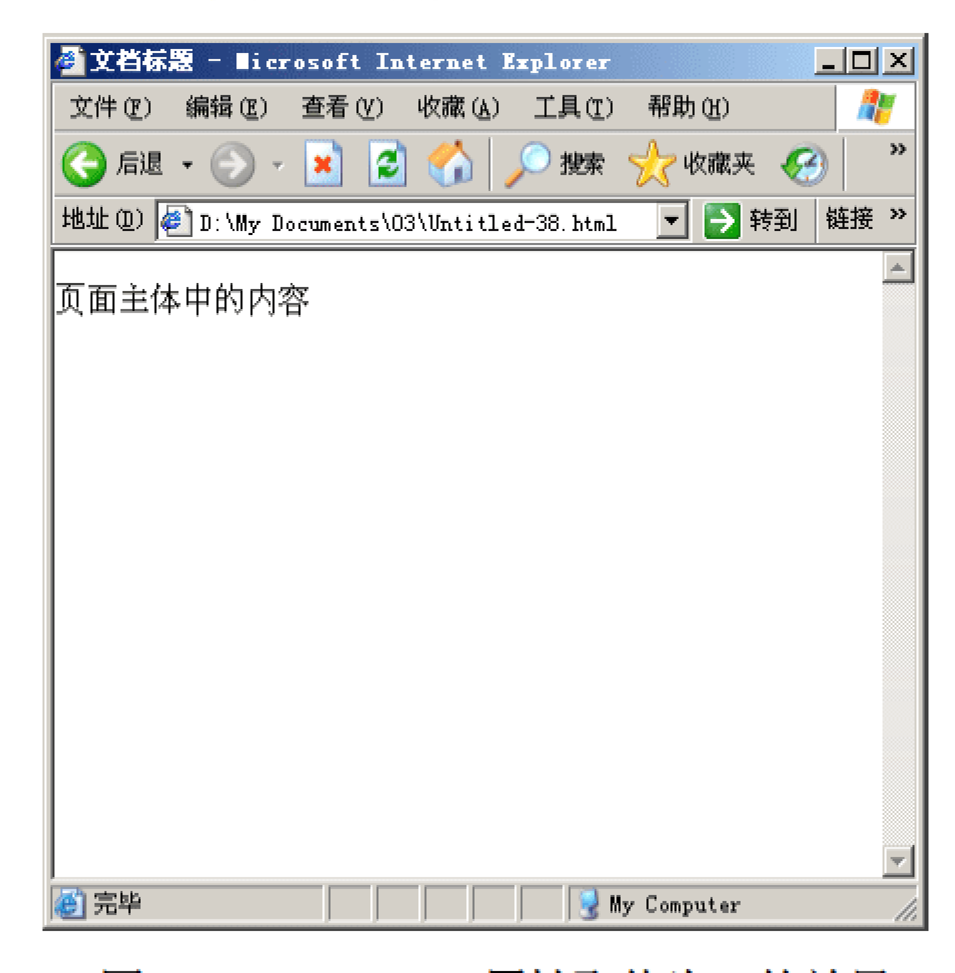


图 3-22 leftmargin 属性取值为 0 的效果

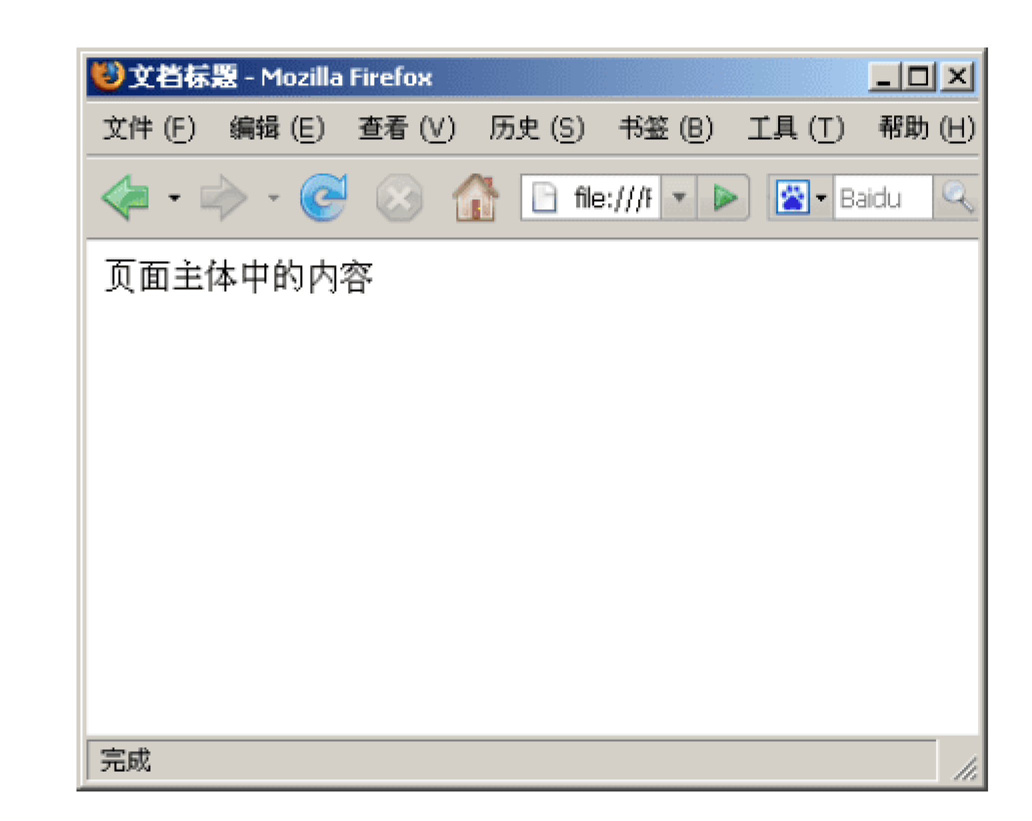


图 3-23 leftmargin 属性取值为 0 时 Firefox 中的效果

3.9.6 IE 浏览器中的上边界属性 topmargin

topmargin 属性用来指定页面和浏览器上边界之间的距离。语法结构如下所示。

```
<body topmargin ="数字值">包含的内容部分</body>
```

在 topmargin 属性中，数字值的实际单位也是像素。下面是 topmargin 属性的实例，其代码如下所示。

例程 3-37 body-topmargin.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>文档标题</title>
```

```
07  </head>
08  <body topmargin ="50">
页面主体中的内容
09  </body>
10  </html>
```

在实例中，08 行定义了页面与浏览器左侧的距离为 50。运行后，在 IE 浏览器中的显示效果如图 3-24 所示。该实例在 Firefox 浏览器中的显示效果如图 3-25 所示。

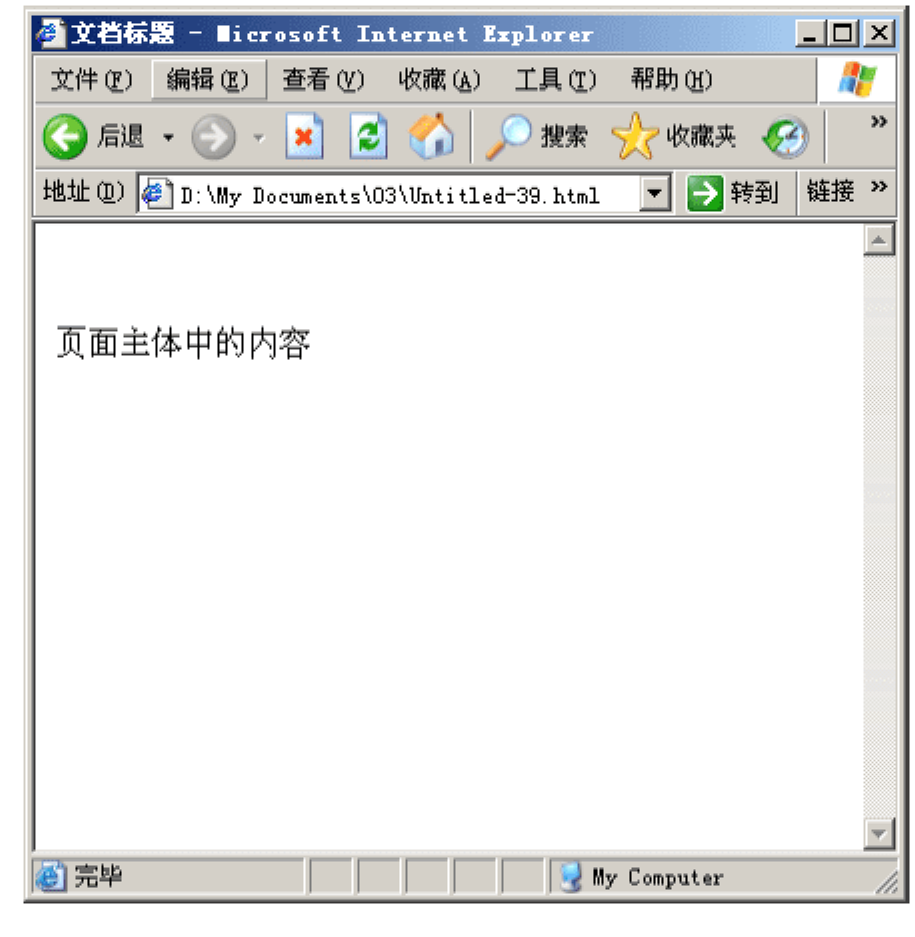


图 3-24　topmargin 属性取值为 50 的效果



图 3-25　topmargin 属性取值为 50 时 Firefox 中的效果

3.9.7　未访问过的链接属性 link

页面主体元素的链接属性，用来定义页面中含有链接的文本的颜色。链接属性中包括 3 个属性：link 属性、vlink 属性和 alink 属性。其中，每个值对应着链接的不同状态。

link 属性用来指定未访问过的链接的颜色。语法结构如下所示。

```
<body link ="颜色值">包含的链接部分</body>
```



注意

link 属性定义的文本颜色只对含有链接的文本起作用。

下面是一个使用 link 属性的实例，其代码如下所示。

例程 3-38　body-link.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>文档标题</title>
</head>
```

```
08 <body link="#cccccc" >
   <a href="#">含有链接的内容</a>
09 </body>
10 </html>
```

该实例中，08 行定义链接文本的颜色为浅灰色。其运行后的显示效果如图 3-26 所示。如果取消 link 属性的定义，则显示效果如图 3-27 所示。

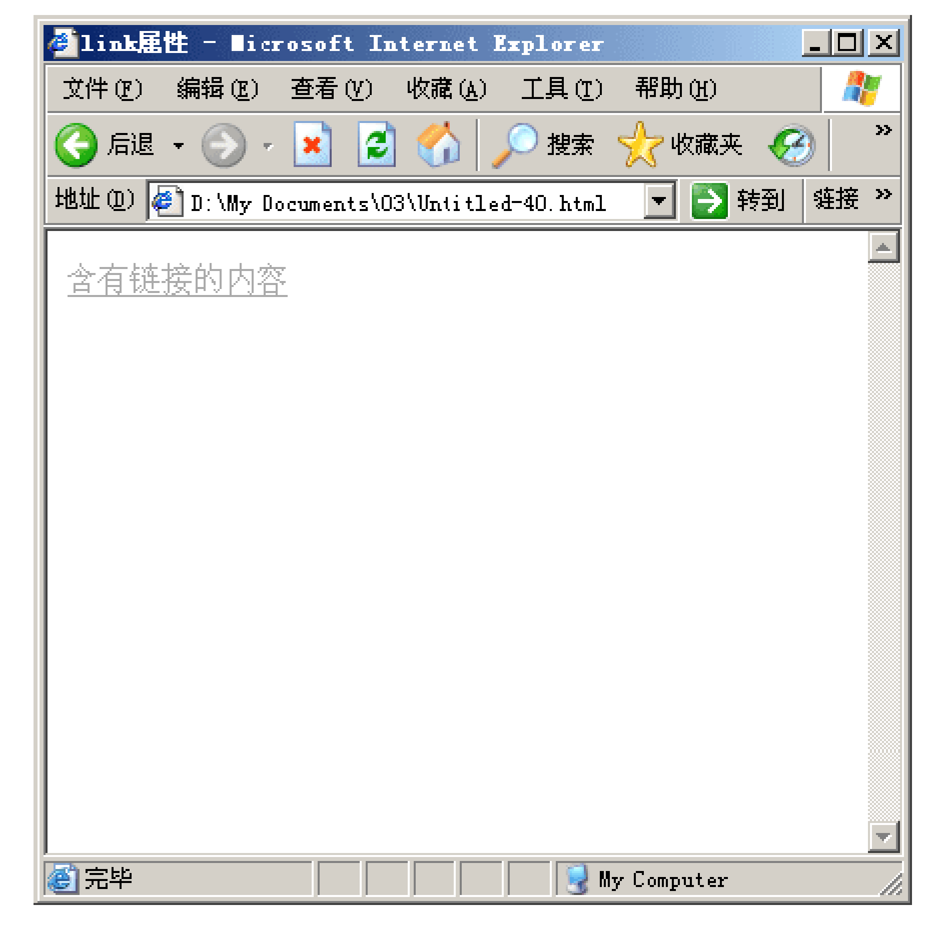


图 3-26　定义 link 属性后的显示效果

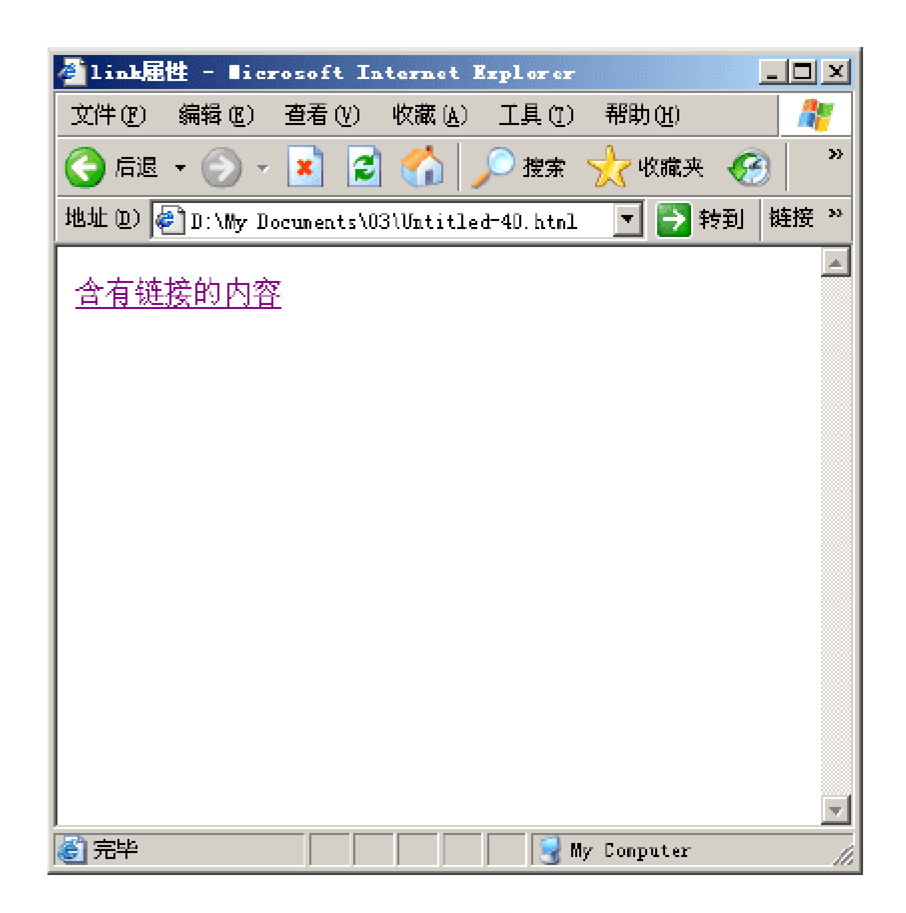


图 3-27　取消 link 属性后的默认效果

3.9.8　已访问过的链接属性 vlink

vlink 属性用来指定已访问过的链接的颜色。语法结构如下所示。

```
<body vlink ="颜色值">包含的链接部分</body>
```

同样，vlink 属性定义的文本颜色，也只对含有链接的文本起作用。下面是一个使用 vlink 属性的实例，其代码如下所示。

例程 3-39　body-vlink.html

```
01 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
02 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
03 <html xmlns="http://www.w3.org/1999/xhtml">
04 <head>
05 <meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
06 <title>文档标题</title>
07 </head>
08 <body vlink="#CCCCCC" >
<a href="#">含有链接的内容</a>
09 </body>
10 </html>
```

该实例中，08 行定义已经访问过的链接文本的颜色为浅灰色。其运行后，当链接被访问后的显示效果，如图 3-28 所示。如果取消 vlink 属性的定义，则显示效果如图 3-29 所示。

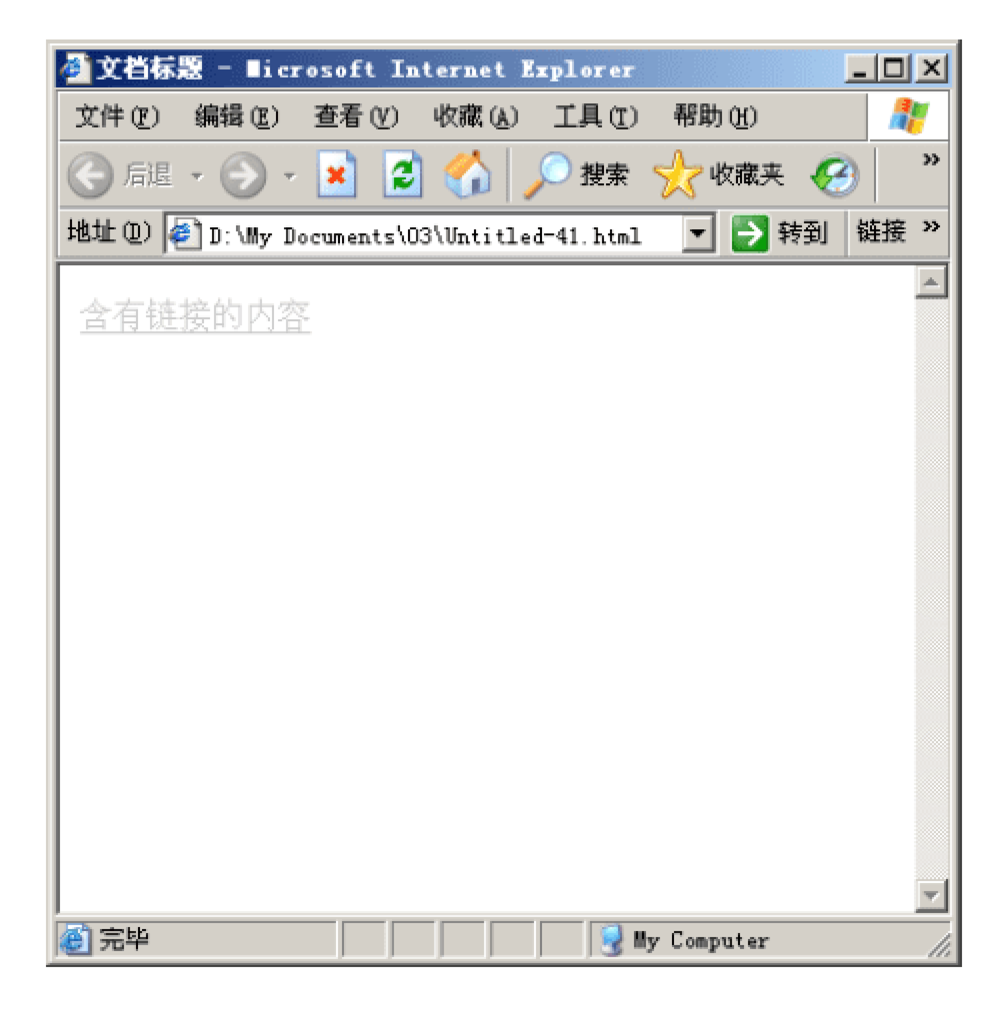


图 3-28 定义 vlink 属性后的显示效果

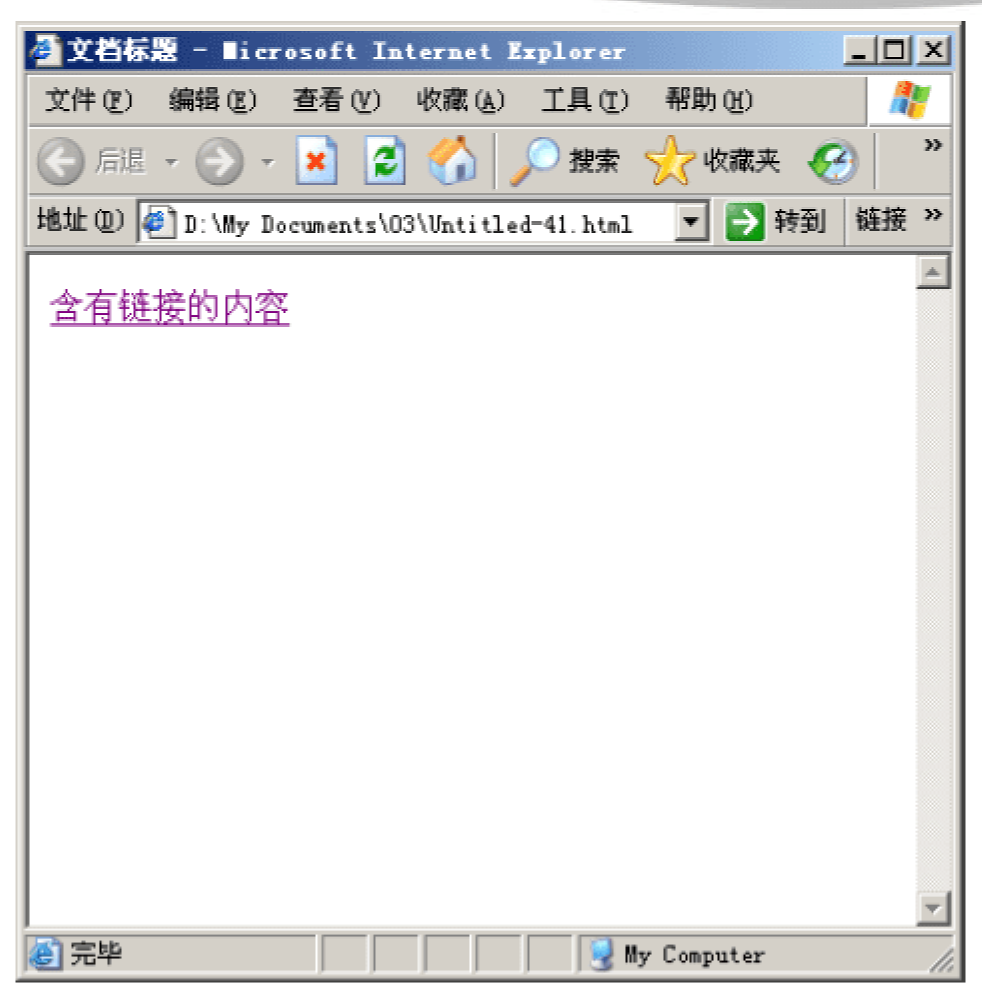


图 3-29 取消 vlink 属性后的默认效果

3.9.9 激活的链接属性 alink

alink 属性用来指定激活的链接文本的颜色，如在已访问的链接上按下鼠标后的状态等。其语法结构如下所示。

```
<body alink ="颜色值">包含的链接部分</body>
```

同样，alink 属性定义的文本颜色，也只对含有链接的文本起作用。下面是一个使用 alink 属性的实例，其代码如下所示。

例程 3-40 body-alink.html

```
01 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
02 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
03 <html xmlns="http://www.w3.org/1999/xhtml">
04 <head>
05 <meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
06 <title>文档标题</title>
07 </head>
08 <body alink="#999999" >
<a href="#">含有链接的内容</a>
09 </body>
10 </html>
```

该实例中，08 行定义已经访问过的链接文本的颜色为灰色。其代码运行后，在已访问链接文本上，单击鼠标后的显示效果，如图 3-30 所示。如果取消 alink 属性的定义，则显示效果如图 3-31 所示。

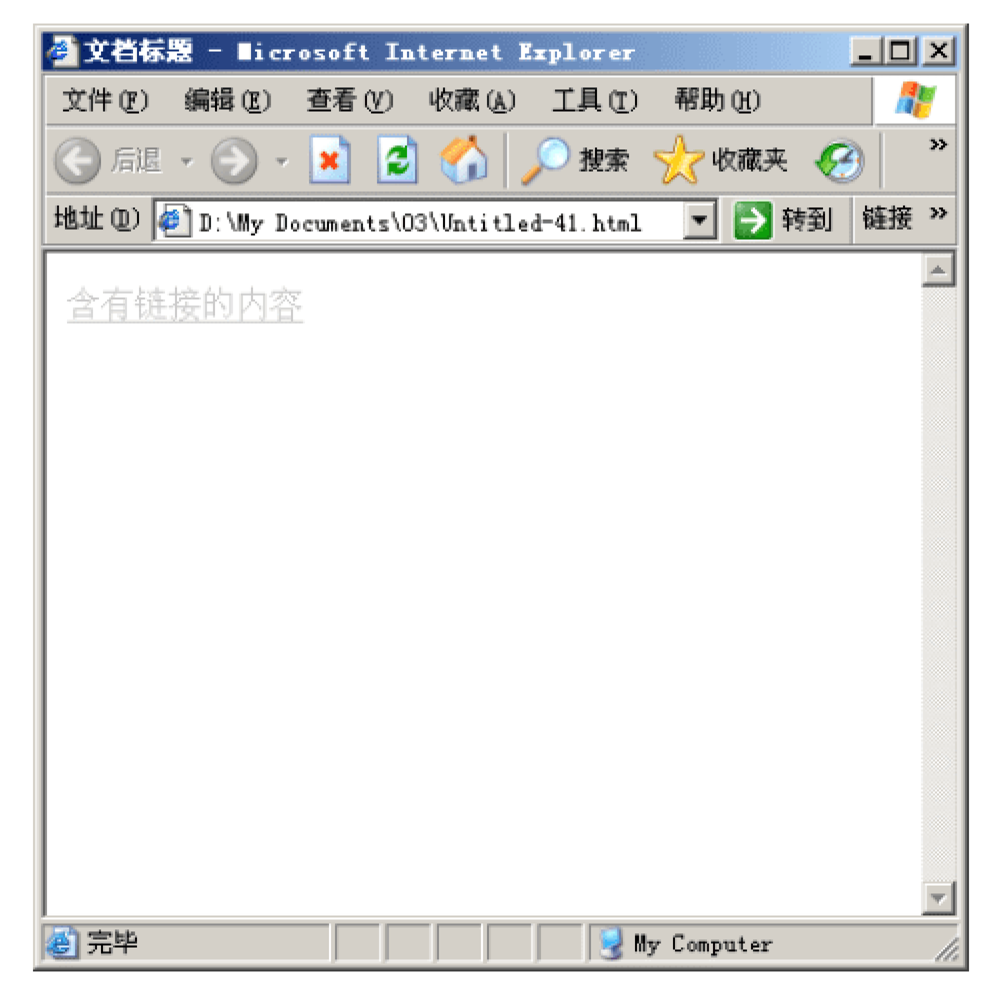



图 3-30　定义 alink 属性后的显示效果

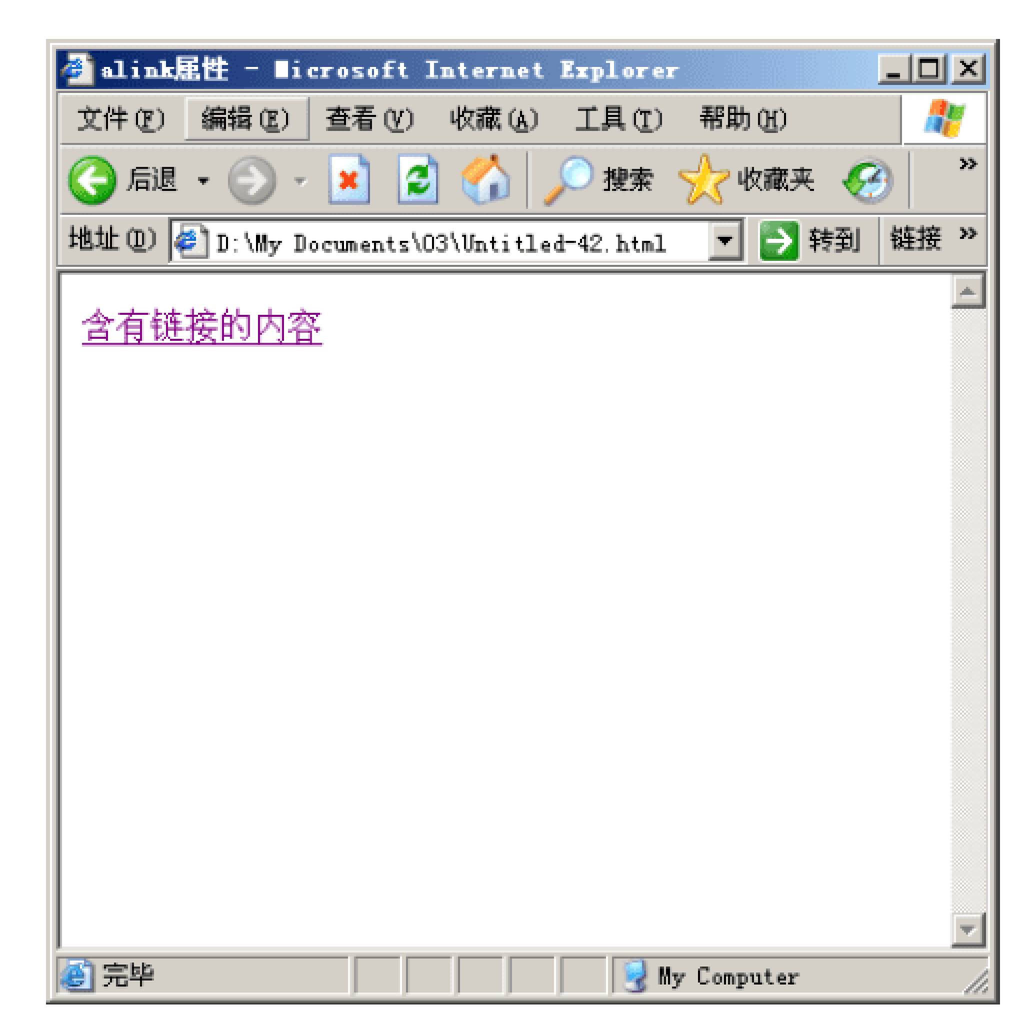


图 3-31　取消 alink 属性后的默认效果

3.9.10　主体元素中使用样式属性 style

页面主体元素的样式属性，用来给页面内容定义级联样式表。其中包括 style 属性、class 属性。下面进行详细讲解。

style 属性用来定义页面主体元素中使用的级联样式表。语法结构如下所示。

```
<body style="级联样式表"></body>
```

关于级联样式表的语法和详细内容，请参见本书第 11 章。下面通过一个使用 style 属性的实例。其代码如下所示。

例程 3-41　body-style.html

```
01 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
02 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
03 <html xmlns="http://www.w3.org/1999/xhtml">
04 <head>
05 <meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
06 <title>style 属性</title>
07 </head>
08 <body style="font-size:24px; font-weight:bold; color:#666666;">
大江东去浪淘尽，千古风流人物。
09 </body>
10 </html>
```

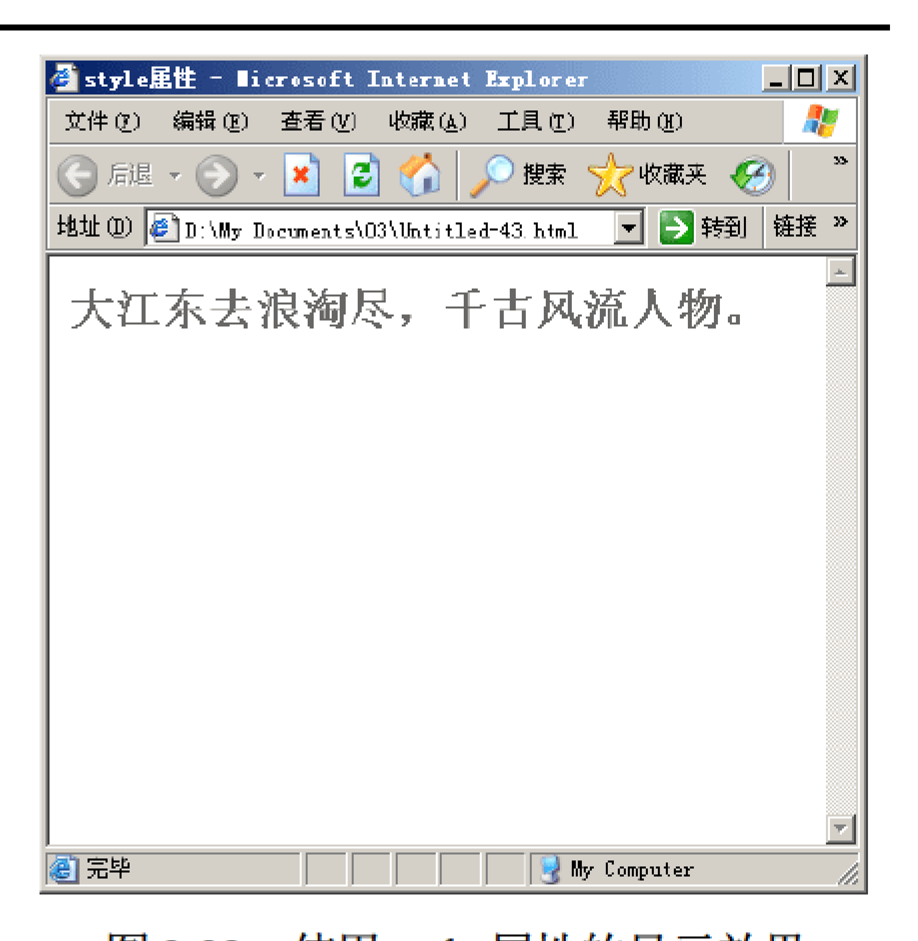


图 3-32　使用 style 属性的显示效果

在以上代码中，08 行使用 style 属性定义了文本的字体大小为 24 像素，字体的样式为加粗，字体的颜色为灰色。其运行后的显示效果如图 3-32 所示。

3.9.11 主体元素中调用样式属性 class

class 属性用来调用级联样式表。使用 class 属性，既可以调用页面头部<style>元素中定义的样式表，也可以调用使用<link>元素链接的外部样式表。其语法结构如下所示。

```
<body class="定义的类的名称"></body>
```

下面介绍一个使用 class 属性的实例。其代码如下所示。

例程 3-42 body-class.html

```
01  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
02  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
03  <html xmlns="http://www.w3.org/1999/xhtml">
04  <head>
05  <meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
06  <title>文档标题</title>
07  <link href="main.css" rel="stylesheet" />
08  </head>
09  <body class="main.css">
页面内容部分，注意文本的显示效果。
10  </body>
11  </html>
```

该实例中，09 行使用 class 属性调用了一个名称为 main.css 的外部级联样式文件。在 main.css 中定义的具体样式如下所示。

```
body{
    background: #CCCCCC;
    color:#ffffff;
    font-size:24px;
    font-weight:bold;}
```

在该样式中，定义了页面中字体的颜色为白色，背景颜色为灰色，字体大小为 24 像素，字体的样式为加粗。其运行后的显示效果如图 3-33 所示。

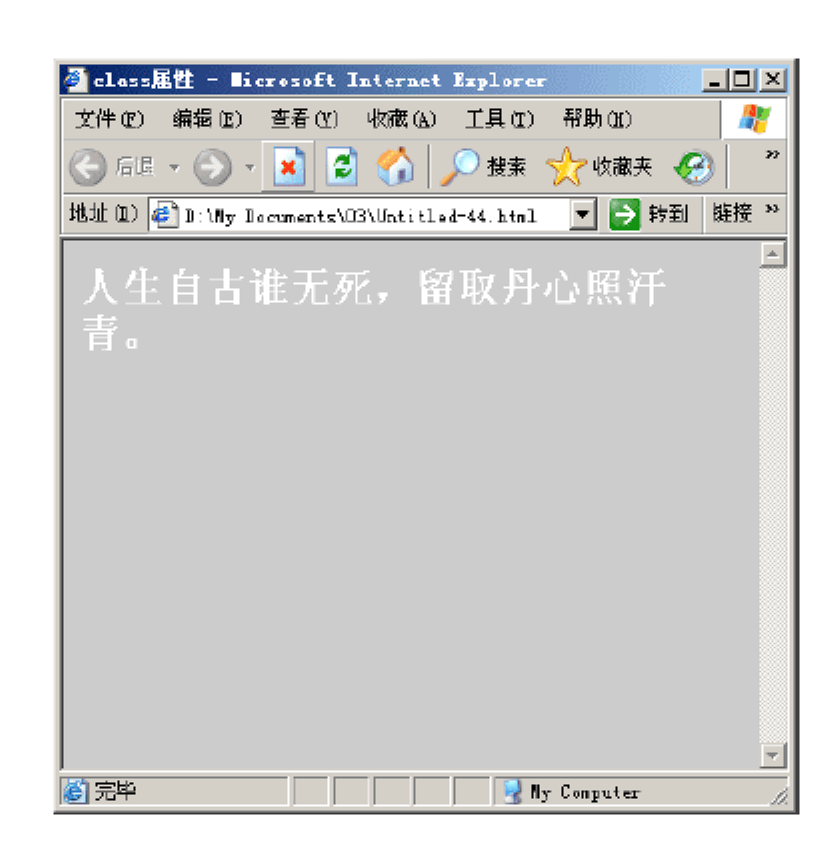


图 3-33 使用 class 属性的显示效果

3.10 使用背景音乐

页面中添加背景音乐的元素是<bgsound>，其中包括音乐地址属性 src、重复属性 loop。<bgsound>属性是 IE 浏览器的特有属性。下面进行详细讲解。

3.10.1 IE 浏览器中添加背景音乐元素<bgsound>

<bgsound>元素的作用是：制作页面的背景音乐。其语法结构如下所示。

```
<bgsound 属性="属性值" />
```

下面是使用<bgsound>元素的实例。其代码如下所示。

例程 3-43 bgsound.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>赤壁怀古</title>
</head>
<body>
<bgsound src="music.mp3" loop="2" />
大江东去，浪淘尽，千古风流人物。 故垒西边，人道是、三国周郎赤壁。 乱石穿空，惊涛拍岸，卷起千堆雪。 江山如画，一时多少豪杰！
</body>
</html>
```

该实例的代码运行后，页面会播放背景音乐。背景音乐的添加，对页面的显示效果并无影响。但是一般背景音乐的大小会比较大。如果网速不够快，可能会影响第一次播放的质量，所以最好选用尽量小的音乐格式。添加背景音乐后，页面的显示效果如图 3-34 所示。

注意

并不是所有浏览者都喜欢页面的背景音乐。采用<bgsound>元素添加的背景音乐，用户无法控制，所以要谨慎使用。

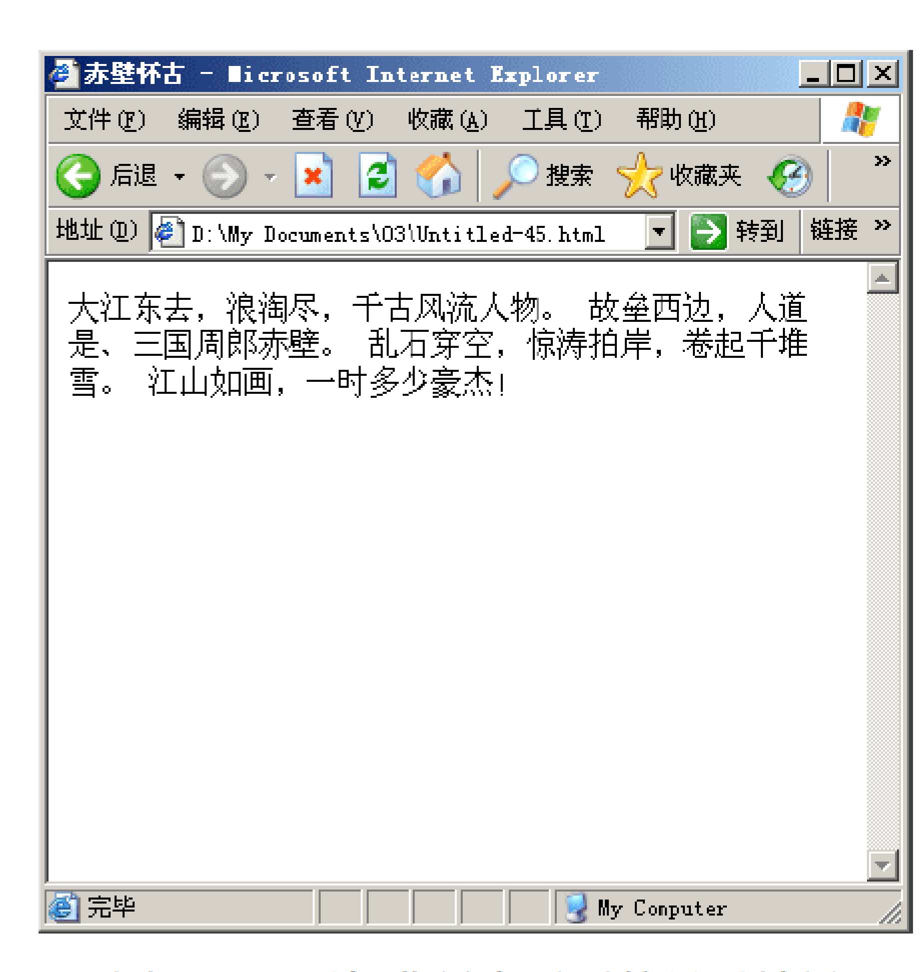


图 3-34 添加背景音乐后的显示效果

3.10.2 背景音乐的路径属性 src

src 属性的作用是：指定页面背景音乐的路径。其语法结构如下所示。

```
<bgsound src="背景音乐的路径" />
```

在<bgsound>元素中，可以支持多种音乐格式，包括常用的 mp3、rm、wav 等。在例程 3-43 中已经使用，这里不再做实例演示。

3.10.3 背景音乐的重复属性 loop

loop 属性指定页面背景音乐的播放次数。语法结构如下所示。

```
<bgsound src="背景音乐的路径" loop="重复次数"/>
```

在页面中如果不设置 loop 属性，背景音乐会不停地重复播放。在例程 3-43 中已经使用 loop 属性，这里不再做实例演示。

3.11 本章习题

一、选择题

1．以下标记符中，用于设置页面标题的是(　　)。

A. <title>　B. <caption>　C. <head>　D. <html>。

2．以下标记符中，没有对应的结束标记的是(　　)。

A.<body>　B.
　C.<html>　D.<title>

3．在网页中，必须使用(　　)标记来完成超级链接。

A.<a>…　B.<p>…</p>　C.<link>…</link>　D.
…

4．若要循环播放背景音乐 bg.mid，以下用法中，正确的是(　　)。

A. <bgsound src="bg.mid" Loop="1">

B. <bgsound src="bg.mid" Loop=True>

C. <sound src="bg.mid" Loop="True">

D. <Embed src="bg.mid" autostart=true></Embed>

5．若要在网页中插入样式表 main.css，以下用法中，正确的是(　　)。

A. <Link href="main.css" type=text/css rel=stylesheet>

B. <Link Src="main.css" type=text/css rel=stylesheet>

C. <Link href="main.css" type=text/css>

D. <Include href="main.css" type=text/css rel=stylesheet>

6. 若要在当前网页中定义一个独立类的样式 myText，使具有该类样式的正文字体为 Arial，字体大小为 9pt，行间距为 13.5pt，以下定义方法中，正确的是(　　)。

A. <Style>

.myText{Font-Familiy:Arial;Font-size:9pt;Line-Height:13.5pt}

</style>

B. .myText{Font-Familiy:Arial;Font-size:9pt;Line-Height:13.5pt}

C. <Style>

.myText{FontName:Arial;FontSize:9pt;LineHeight:13.5pt}

```
</style>
```

D.
```
<Style>
. .myText{FontName:Arial;Font-ize:9pt;Line-eight:13.5pt}
</style>
```

二、填空题

1．文件头标签也就是通常所见到的___标签。

2．创建一个 HTML 文档的开始标记符是________；结束标记符是________。

3．设置文档标题以及其他不在 WEB 网页上显示的信息的开始标记符是________；结束标记符是________。

4．设置文档的可见部分开始标记符是________；结束标记符是________。

5．网页标题会显示在浏览器的标题栏中，则网页标题应写在开始标记符________和结束标记符________之间。

6．要设置一条 1 像素粗的水平线，应使用的 HTML 语句是________。

7．要设置网页在黑色背景下显示白色文字，应使用____________语句。

8．要设置一条 1 像素粗、200 像素长的左对齐的水平线，应使用____________语句。

三、实战练习

1．使用链接路径属性 href 链接图片，可以使用自己电脑的图片，也可以使用网络图片。

2．使用样式元素<style>修饰页面的文字样式，熟悉其中各个属性的意义。

3．制作一个页面，并添加自己喜欢的音乐。

Chapter 4

文本和段落元素

网页中的信息主要通过文本内容(同时辅助适当的图片和多媒体)进行传递。所以文本内容显示的效果，是决定网页成功与否的关键。合理地使用文本元素和相关属性，可以大大提高内容的可阅读性，使制作的页面更有亲和力。

本章主要内容有:

◎ 用段落文本的层元素控制段落文本的对齐、显示位置，样式等。

◎ 用文本标题元素控制标题文本的大小、加粗方式等。

◎ 用文本中文字格式的元素控制内容中文字的显示大小，显示方式等。

◎ 用文本的换行、格式和间隔的元素分隔文本内容，或者设置文本格式。

4.1 层元素<div>

在页面中层元素<div>用来将页面内容分割成各个独立的部分。在每个<div>元素中，不仅可以包含文本内容，也可以包含图片、表单等其他内容。在默认的情况下，<div>元素所包含的内容，将在新的一行显示。其语法结构如下所示。

```
<div>……</div>
```

下面是一个使用<div>元素的实例。其代码如下所示。

例程 4-1　div.html

```
01  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
02  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
03  <html xmlns="http://www.w3.org/1999/xhtml">
04  <head>
05  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
06  <title>层元素</title>
07  </head>
08  <body>
上面是层元素以外的文本
09  <div>这里是层元素的内容部分</div>
注意层元素元素以外的文本的显示方式。
10  </body>
11  </html>
```

该实例中，在 09 行中插入了一个<div>元素。由于<div>元素的插入，被分隔的文本将不能与<div>元素同行显示。其运行后的显示效果，如图 4-1 所示。



图 4-1　<div>元素的默认显示效果

<div>元素中可以使用的所有属性如表 4-1 所示。

表 4-1　<div>元素的所有属性

名　　称	属 性 代 码
文本显示方向属性	dir
指定语言属性	lang
类属性	class
定义级联样式属性	style
对齐属性	align
标题属性	title
取消自动换行属性	nowrap
标记属性	id

4.1.1　标记属性 id

id 属性的作用可以分为两个方面。其一也是最主要的作用是用来标记元素，也就是给元素定义唯一的标识，方便在元素中使用行为。另一个作用是类似 class 属性的作用，用来调用级联样式表。其语法结构如下所示：

```
<div id="定义的名称">……</div>
```

下面是一个使用 id 属性的实例。其代码如下所示。

例程 4-2　div-id.html

```
01  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
02  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
03  <html xmlns="http://www.w3.org/1999/xhtml">
04  <head>
05  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
06  <title>id 属性的应用</title>
07  <link href="main.css" rel="stylesheet" type="text/css" />
08  <style>
09  #body{
      width:200px;
      height:100px;
      border:6px solid #00FF00;
      font-size:26px;
        font-style:italic;}
10  </style>
11  </head>
12  </html>
```

该实例中，07 行中调用了一个名称为 main.css 的外部样式文件。在这个调用的样式文件中，具体定义的样式内容的代码如下所示。

```
body{
    width:200px;
```

```
        height:100px;
        border:6px solid #00FF00;
        font-size:26px;
        font-style:italic;}
```

注意

使用 id 属性调用的样式，其写法与 class 属性调用的样式的选择符有所区别。要在样式前加“#”号(关于选择符的具体内容，请参看第 12 章)。

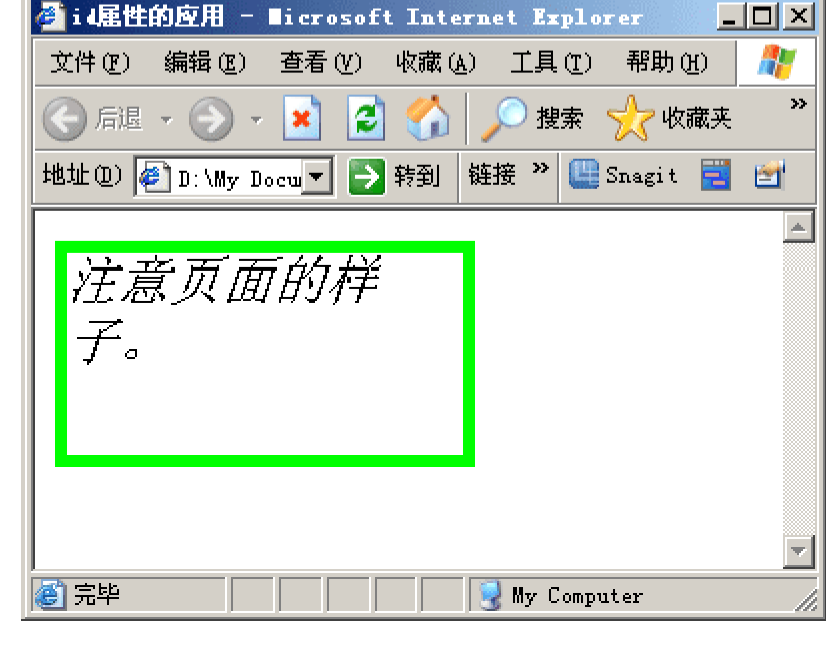


图 4-2　定义 style 属性后的<div>元素

在该样式中，定义的元素的宽度为 200 像素，高度为 100 像素，边框为绿色实线，字体大小为 26 像素，文本样式为斜体。代码运行后，其显示效果如图 4-2 所示。

关于 id 属性的另一个作用，将在第 18 章中进行讲解。

4.1.2　调用样式属性 class

class 属性用来在元素中调用级联样式表。与 body 中的 class 属性的区别在于属性的作用范围不同。其原因在于部分样式表属性具有继承性(关于样式的继承性，请参看第 12 章)。

注意

在<div>元素中，用来控制元素表现的属性很少，甚至不能定义元素的高度和宽度。所以要使用级联样式表来控制其表现。

语法结构如下所示：

```
<div class="定义的类的名称">……</div>
```

class 属性中使用的样式，既可以是页面中<style>元素中定义的样式，也可以是从外部文件中调用的样式。建议使用外部调用的样式。下面是一个使用 class 属性的实例。其代码如下所示。

例程 4-3　div-class.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>文档标题</title>
</head>
<body>
<div class="content">
```

```
    页面内容部分。
10  </div>
11  </body>
12  </html>
```

该实例中，07 行调用了一个名称为 main.css 的外部样式文件。在这个调用的样式文件中，具体定义的样式内容如下所示。

```
body{
    width:150px;
   height:90px;
    border:6px solid #333333;
    background:#3366FF;
    color:#CCCCCC;
    font-size:20px;
   font-style:italic;}
```

在该样式中，定义的元素的宽度为 150 像素，高度为 90 像素，边框为深灰色实线，背景蓝色，文本浅灰色，字体大小为 20 像素的斜体字。代码运行后，其显示效果如图 4-3 所示。

从图 4-3 可以看出，通过级联样式表，可以控制元素的所有显示属性。所以在 HTML 标准中，推荐使用样式表，实现页面结构和表现相分离。

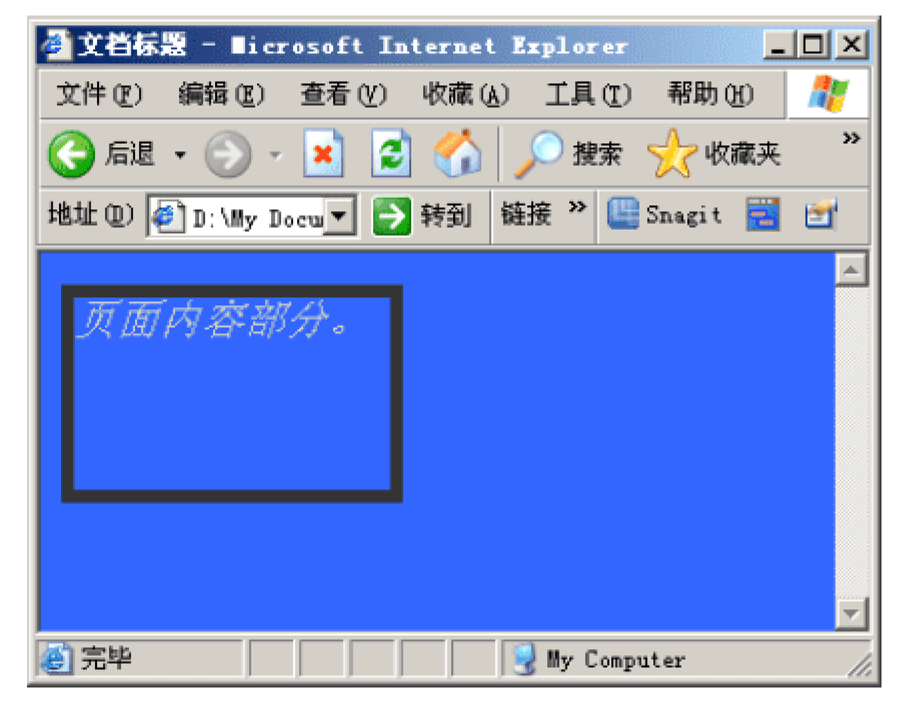


图 4-3　定义 class 属性后的<div>元素

4.1.3　创建样式属性 style

style 属性用来在元素中定义级联样式表。其与 class 属性的区别在于，使用 style 属性定义的样式的优先级，高于 class 属性调用的样式(关于级联样式表中的优先级的详细内容，请参看第 12 章)。语法结构如下所示。

```
<div style="定义的样式">……</div>
```

下面通过一个实例，演示当 class 属性使用外部调用的级联样式表时，页面的显示效果。其代码如下所示。

例程 4-4　div-style.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>样式属性</title>
<div style=" width:300px; height:150px;border:6px solid #000033; font-size:28px;
  text-decoration:line-through;">
  注意页面的显示图像。
</div>
```

```
09  </body>
10  </html>
```

其中 07 和 08 之间的内容是样式属性，在该样式中，定义的元素的宽度为 300 像素，高度为 150 像素，边框为深蓝色实线，字体大小为 26 像素，文本修饰为删除线。代码运行后，其显示效果如图 4-4 所示。

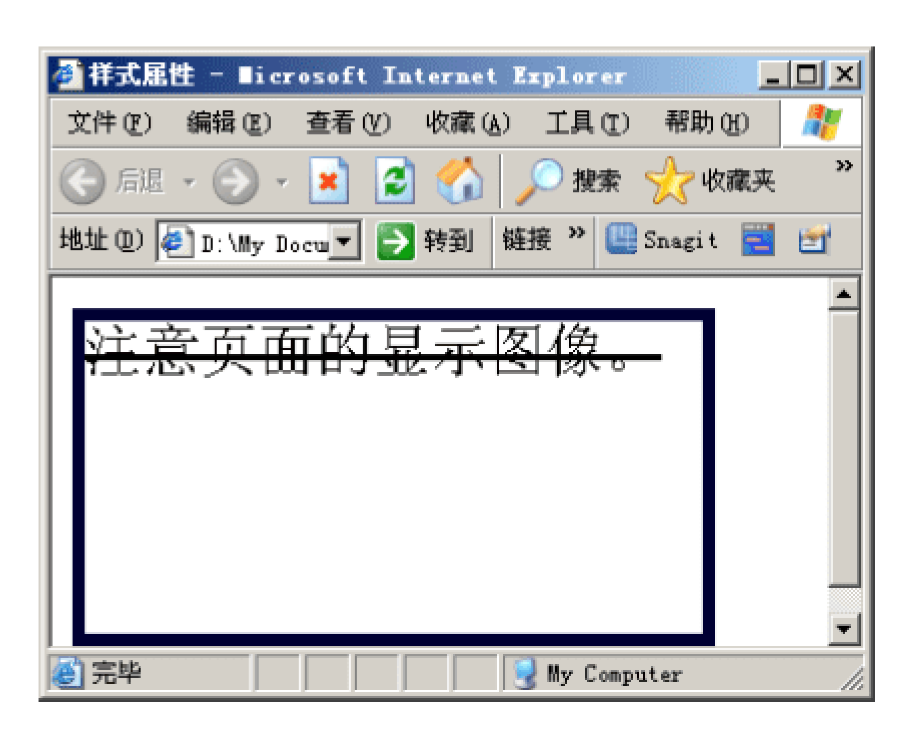


图 4-4　定义 style 属性后的<div>元素

4.1.4　对齐属性 align

对齐属性用来定义内容的水平对齐方式。使用对齐属性，不仅可以控制文本内容的对齐，也可以控制文本中图片等内容的对齐。语法结构如下所示。

```
<div align="水平对齐方式">包含的内容部分</div>
```

对齐属性 align 可以取 4 个值，其每个值的具体含义如表 4-2 所示。

表 4-2　区域元素对齐属性的取值及含义

属 性 代 码	代码的含义
left	元素中内容靠元素左侧对齐(默认值)
center	元素中内容靠中间对齐
right	元素中内容靠元素右侧对齐
justify	元素中内容靠元素两端对齐

下面是一个使用 align 属性的实例。其代码如下所示。

例程 4-5　div-align.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>align 属性</title>
</head>
<body>
<div align="right">
这里是 align 元素的内容部分
<br />
的一段内容
</div>
</body>
</html>
```

说明

因为<div>元素中没有宽度和高度属性，所以为了显示右对齐的效果，在代码中使用了
元素。
元素的含义是，使文本在分隔处换行显示(关于
元素的详细内容，请参见 4.4.1 节)。

该实例中，09 行在对齐属性中使用了 right 值，目的是使其中的内容靠右侧对齐。其运行后的显示效果，如图 4-5 所示。

在对齐属性的 4 个值中，justify 值较为特殊。因为到目前为止，还没有任何浏览器支持此属性值，所以使用 justify 值时，内容会以左对齐的方式显示。下面是 align 属性中使用 justify 值的实例，其代码如下所示。

例程 4-6　div-align-justify.html

```
01  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
02  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
03  <html xmlns="http://www.w3.org/1999/xhtml">
04  <head>
05  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
06  <title>align 属性</title>
07  </head>
08  <body>
09  <div align="justify">
     这里是 align 属性元素的内容部分<br />另一段内容
10  </div>
11  </body>
12  </div>
13  </body>
14  </html>
```

09 行中使用了 justify 属性，该代码运行后，显示效果如图 4-6 所示。

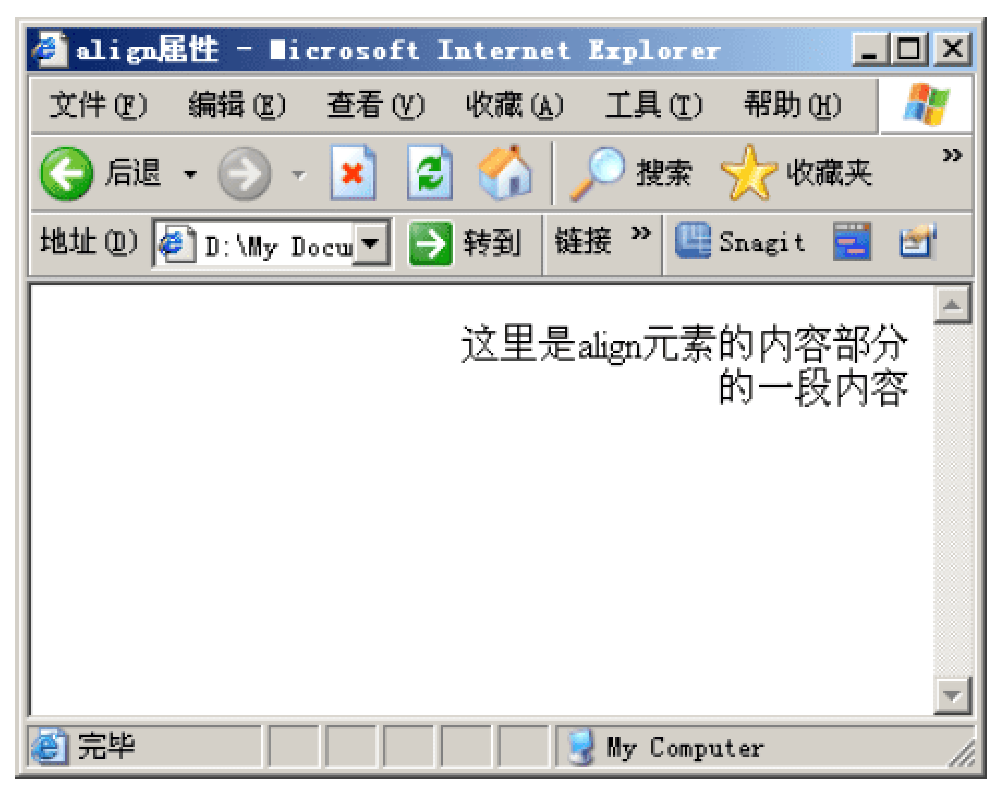



图 4-5　对齐属性取值为 right 时的显示效果

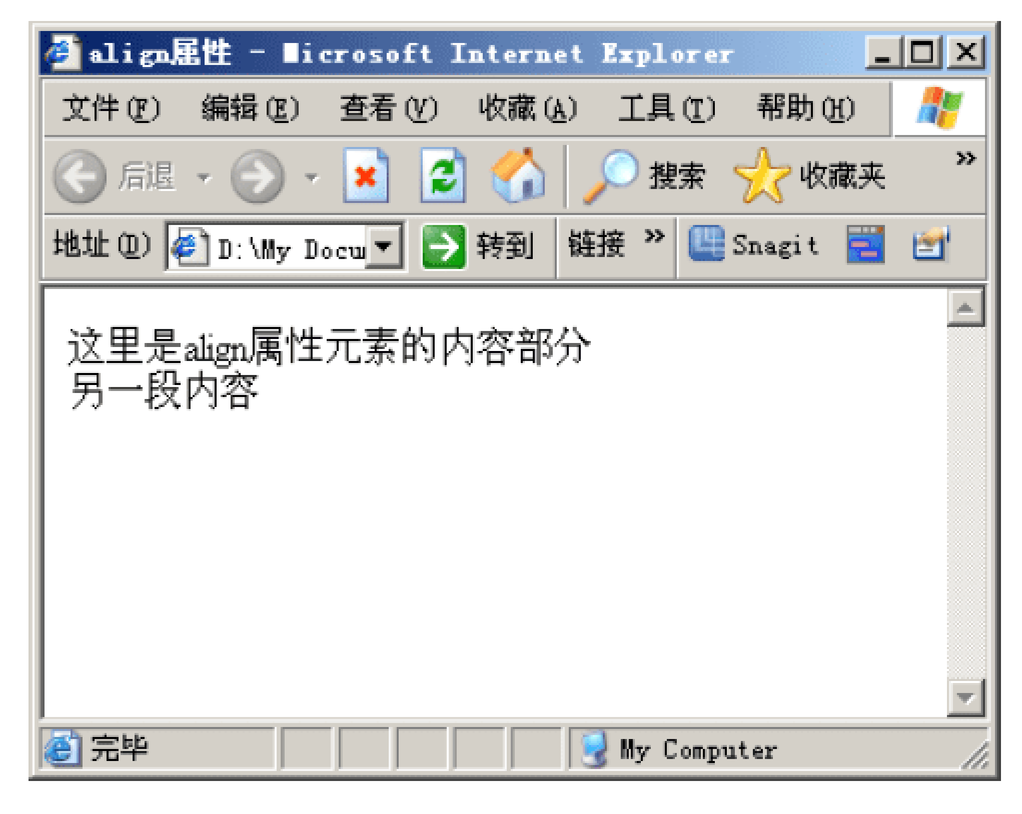


图 4-6　对齐属性取值为 justity 时的显示效果

在 HTML 中，并不建议使用 align 属性控制内容的对齐。推荐的方法是，使用样式表控制内容的对齐。

4.1.5 取消自动换行属性 nowrap

nowrap 属性用来使文本内容同行显示。在 XHTML 中文本的默认显示方式是，忽略掉文本中“回车键”的换行符，根据元素的宽度进行自动换行显示。使用 nowrap 属性可以改变这种显示方式，使文本遇到“回车键”的换行符时换行显示。语法结构如下所示。

```
<div nowrap="nowrap">……</div>
```

nowrap 属性没有值，在 HTML 中使用自身名称作为值。因为大多数浏览器都不支持该属性，所以就不做实例演示了。

4.1.6 标题属性 title

title 属性用来设定当鼠标悬停在文本内容上时，所显示的内容。通过 title 属性，可以给文本添加注释等。目前大多数浏览器并不支持该属性，只有 IE 浏览器可以支持该属性。其语法结构如下所示。

```
<div title="标题内容">……</div>
```

下面是一个使用 title 属性的实例。其代码如下所示。

例程 4-7 div-title.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>文档标题</title>
</head>
<body>
<div title="内容部分的帮助等内容">页面内容部分。</div>
</body>
</body>
</html>
```

其代码运行后，鼠标悬停在文本内容上时，页面中会出现“内容部分的帮助等内容”的句子。在 IE 浏览器中的显示效果，如图 4-7 所示。

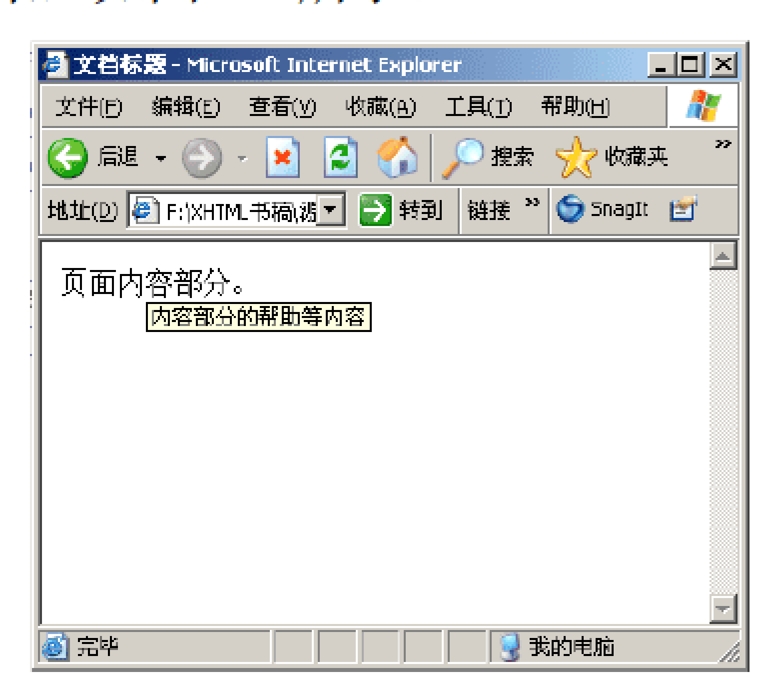


图 4-7 使用 title 属性后的显示效果

4.2 标题元素

标题元素用来定义内容的标题。其中包括 6 个元素，分别是<h1>、<h2>、<h3>、<h4>、<h5>和<h6>，它们分别对应 6 种显示效果。以<h1>元素为例，语法结构如下所示。

```
<h1>……</h1>
```

下面通过一个实例，演示下唐诗《无题》默认情况下的显示方式。其代码如下所示。

例程 4-8 h.html

```
01   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
02   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
03   <html xmlns="http://www.w3.org/1999/xhtml">
04   <head>
05   <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
06   <title>唐诗</title>
07   </head>
08   <body>
    <h1>无题</h1>
    <h2>李商隐</h2>
    <h3>昨夜星辰昨夜风</h3>
    <h4>画楼西畔桂堂东</h4>
    <h5>身无彩凤双飞翼</h5>
    <h6>心有灵犀一点通</h6>
09   </body>
10   </html>
```

该实例的主体元素中，在 08 和 09 行之间定义了 6 个<h>元素。其运行后的显示效果，如图 4-8 所示。

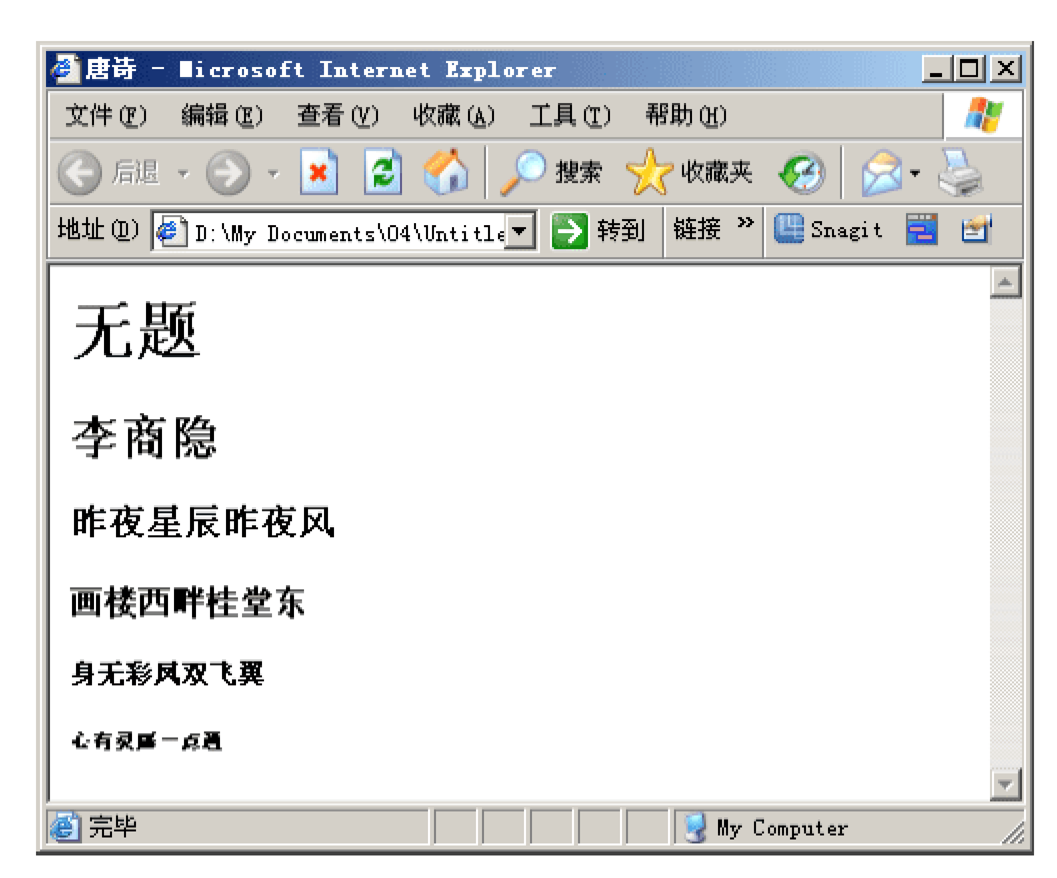


图 4-8 6 个<h>元素的默认显示效果

从图 4-8 可以看出，使用标题元素时，上下元素之间会分隔开一段距离。标题元素中可以使用的所有属性如表 4-3 所示。

表 4-3　标题元素的所有属性

属 性 名 称	代 码 写 法
文本显示方向属性	dir
指定语言属性	lang
类属性	class
定义级联样式属性	style
对齐属性	align
标题属性	title
标记属性	id

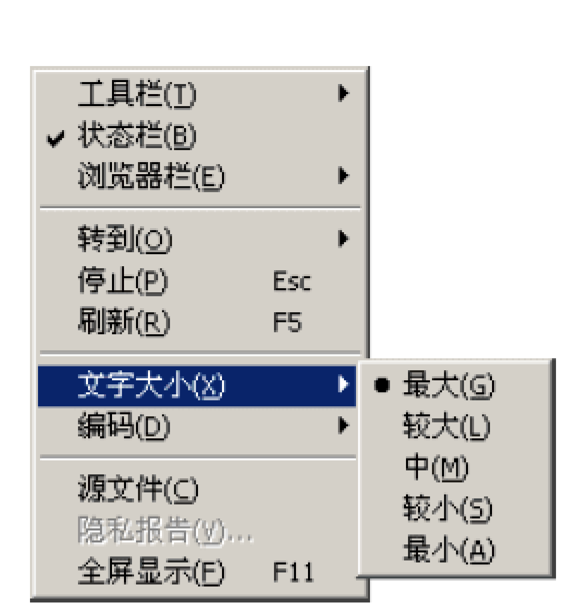
在标题元素可以包含所有内联元素，包括<a>、<img>、<strong>等。在标题元素中使用块元素，由于浏览器的容错性，一般可以正常显示。

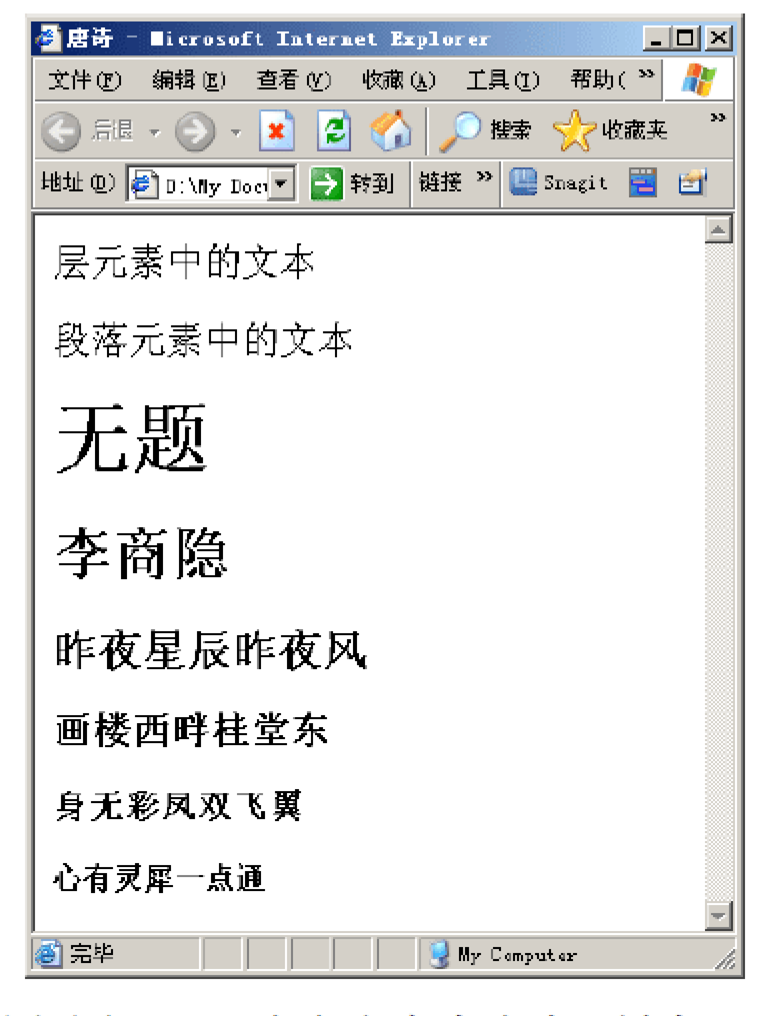
页面中文本显示的大小是受到浏览器的影响的。不但标题元素是如此，包括前面讲解的<div>元素和<p>元素，以及<body>元素中的文本都会受到浏览器设置的影响。如果要使文本大小固定不变，可以使用级联样式表实现。下面是一个使用标题元素的实例。其代码如下所示。

例程 4-9　h-use.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>唐诗</title>
</head>
<body>
<div>层元素中的文本</div>
<p>段落元素中的文本</p>
<h1>无题</h1>
<h2>李商隐</h2>
<h3>昨夜星辰昨夜风</h3>
<h4>画楼西畔桂堂东</h4>
<h5>身无彩凤双飞翼</h5>
<h6>心有灵犀一点通</h6>
  </body>
  </html>
```

该实例中，分别在已经讲解的各种标题元素中定义了文本。下面讲解下浏览器关于字体显示大小的相关设置(以 IE 6.0 为例)。在 IE 6.0 的主菜单栏中，单击“查看”|“文字大小”命令，显示如图 4-9 所示的下拉菜单。在下拉菜单中选择“最大”命令，此时文本的显示效果如图 4-10 所示。选择“最小”命令，此时文本的显示效果如图 4-11 所示。





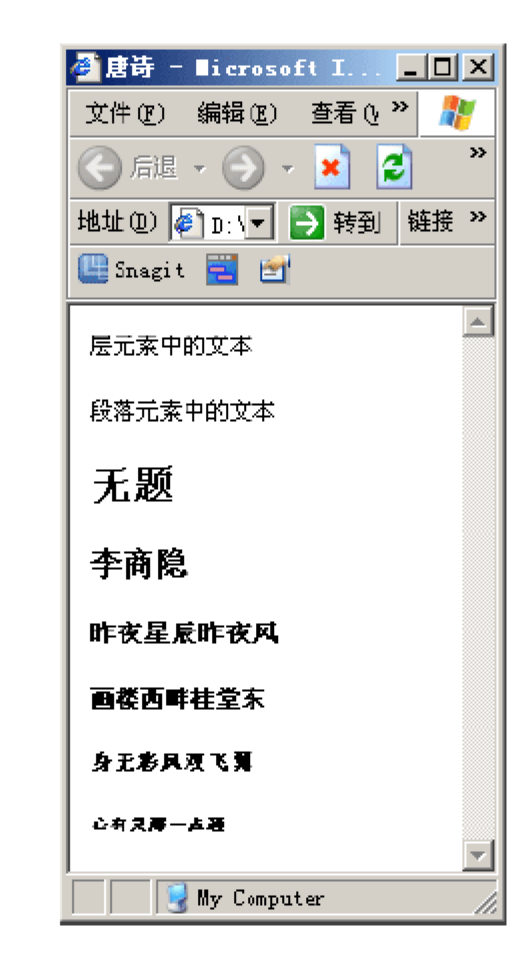


图 4-9 浏览器中文字大小的设置　图 4-10 定义文字大小为“最大”　图 4-11 定义文字大小为“最小”

4.3 段落元素<p>

段落元素<p>用来定义一个段落。在<p>元素中，可以包含文本、图片，以及用来修饰文本的元素，如<strong>元素等。和<div>元素一样，被<p>元素包含的内容，默认的显示方式是换行显示。语法结构如下所示。

```
<p>……</p>
```

下面是一个使用<p>元素的实例。其代码如下所示。

例程 4-10　p.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>段落元素的应用</title>
</head>
<body>
段落元素以外的文本显示方式
<p>这里是段落元素的内容部分</p>
注意段落元素以外的文本的显示方式。
</body>
</html>
```

该实例中，在 09 行中插入了一个<p>元素。由于<p>元素的插入，被分隔的文本，将不能与<p>元素同行显示。其运行后的显示效果，如图 4-12 所示。

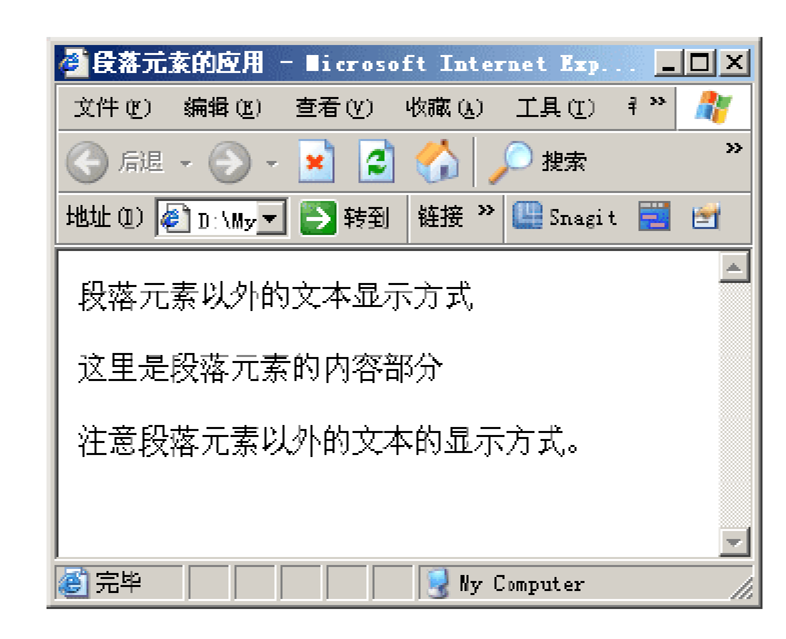


图 4-12　<p>元素的默认显示效果

从图 4-12 可以看出，<p>元素和<div>元素在显示上的区别。使用<div>元素包含的内容，和元素外分隔的文本之间是不存在距离的。而使用<p>元素时会使各行之间分隔开一段距离。<p>元素中可以使用的所有属性如表 4-4 所示。

表 4-4　<p>元素的所有属性

属性名称	写　法
文本显示方向属性	dir
指定语言属性	lang
类属性	class
定义级联样式属性	style
对齐属性	align
标题属性	title
标记属性	id

4.4 文本的间隔和布局

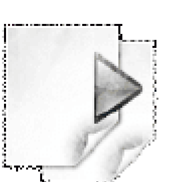
文本的间隔和布局主要是指：在页面中用来区分文本内容的元素，还有控制内容显示方式的元素。其中包括
、<pre>等元素。其具体用法如下所示。

4.4.1 换行元素

换行元素
用来使被分隔的文本换行显示。语法结构如下所示。

```
<br />
```

下面是使用
元素的实例。其代码如下所示。



注意

从本节以后，本书只写出主要的代码，大家运行的时候请注意。

例程 4-11 br.html

```
<body>
无题 <br />李商隐 <br />春蚕到死丝方尽 <br />蜡炬成灰泪始干
</body>
```

代码
运行后，虽然写在一行，但是它们在页面换自动换行。显示效果如图 4-13 所示。
元素中可以使用的所有属性，如表 4-5 所示。

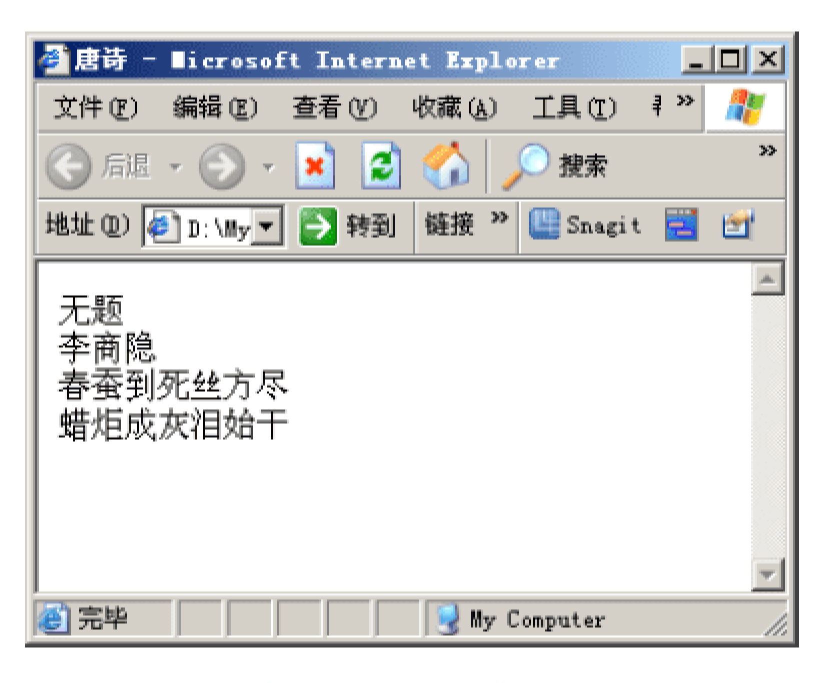


图 4-13　使用
元素的显示效果

表 4-5　
元素的所有属性

属 性 名 称	写　法
清除属性	clear
类属性	class
定义级联样式属性	style
标记属性	id
标题属性	title

元素里还有一个最重要的属性，就是清除属性 clear，clear 属性用来使被分隔的文本不能与其他元素同行显示。例如，在文本的某一侧含有图片或者其他元素，当换行符中使用 clear 属性时，换行后的文本将在图片或者其他元素的下面显示。语法结构如下所示。

```
<br clear="属性值" />
```

clear 属性可以使用 3 个值，其具体含义如表 4-6 所示。

表 4-6　clear 属性的取值及含义

属　性　值	代表的含义
left	清除左侧元素
all	清除两边元素
right	清除右侧元素
none	不清除元素

下面是使用
元素的clear属性的实例。其代码如下所示。

例程 4-12 br-clear.html

```
01 <body>
02 <img src="http://www.baidu.com/img/baidu_logo.gif" align="left" alt="logo" />文本内容<br clear="left" />换行符分隔后的文本。
03 </body>
```

其代码运行后，显示效果如图 4-14 所示。在该实例中，02 行中<img>元素定义了相应的对齐属性，关于<img>元素的详细内容请参看第 6 章。



图 4-14　使用 clear 属性的显示效果

4.4.2 缩进元素<blockquote>

缩进元素<blockquote>用来使包含的内容从文本中分离出来，同时首尾缩进显示。语法结构如下所示。

```
<blockquote>……</blockquote>
```

下面是使用<blockquote>元素的实例。其代码如下所示。

例程 4-13 blockquote.html

```
01 <body>
唐诗是我国历史文化的最优秀部分<blockquote>唐代是我国封建社会中经济政治高度发展的第一个高峰</blockquote>从而除了一批优秀的诗人
02 </body>
```

这是一段比较长的文本内容，注意文本中缩进元素<blockquote>所包含的内容的显示方式，显示效果如图 4-15 所示。



图 4-15 使用<blockquote>元素的显示效果

<blockquote>元素中可以使用的所有属性如表 4-7 所示。

表 4-7 <blockquote>元素的所有属性

属 性 名 称	代 码 写 法
文本显示方向属性	dir
指定语言属性	lang
类属性	class
定义级联样式属性	style
标记属性	id
标题属性	title
引用地址属性	cite

4.4.3 保留格式元素<pre>

保留格式元素<pre>用来使包含的内容按照文档源代码的格式显示。因为浏览器的默认显示方式中，将压缩多个空格为一个，同时忽略换行等空白符号。其语法结构如下所示。

```
<pre>……</pre>
```

下面是使用<pre>元素的实例。其代码如下所示。

例程 4-14 pre.html

```
01 <body>
```

```
<pre>注意页面的显示方式
无题
唐        李商隐</pre>
</body>
```

代码运行后，显示效果如图 4-16 所示。

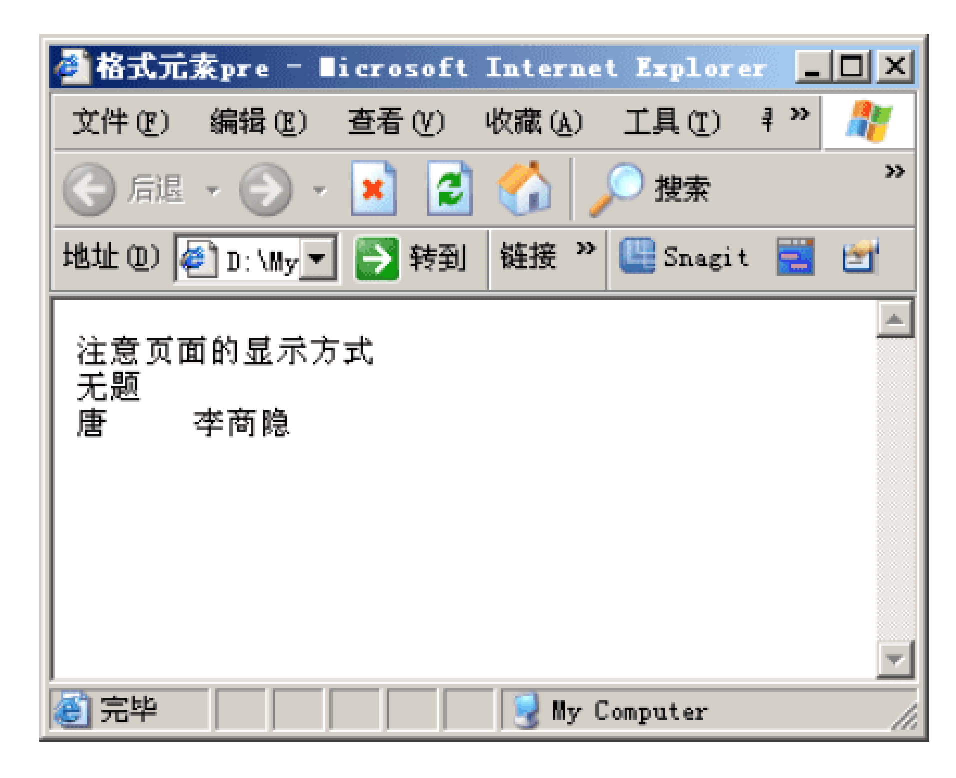


图 4-16　使用<pre>元素的显示效果

<pre>元素中可以使用的所有属性如表 4-8 所示。

表 4-8　<pre>元素的所有属性

属 性 名 称	写　法
文本显示方向属性	dir
指定语言属性	lang
类属性	class
定义级联样式属性	style
标记属性	id
标题属性	title
宽度属性	width

4.4.4　取消换行元素<nobr>

取消换行元素<nobr>用来使被包含的文本同行显示。其语法结构如下所示。

```
<nobr>……</nobr>
```

下面是使用<nobr>元素的实例。其代码如下所示。

例程 4-15　nobr.html

```
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>唐诗</title>
<style>
content{
```

```
    width:260px;
        height:130px;
        border:6px solid #666666;}
06  </style>
07  </head>
08  <body>
09  <div class="content"><nobr>注意页面内容的显示方式。白日依山尽，黄河入海流</nobr></div>
10  </body>
11  </html>
```

在该实例中，09 行中使用了<div>元素，并且通过调用级联样式表，定义其宽度为 260 像素，高度为 130 像素，边框为深灰色实线。目的是使<nobr>元素的文本效果显示得更明显。其代码运行后，所有的文字都不会换行，显示效果如图 4-17 所示。如果取消<nobr>元素，其代码运行后，显示效果如图 4-18 所示。

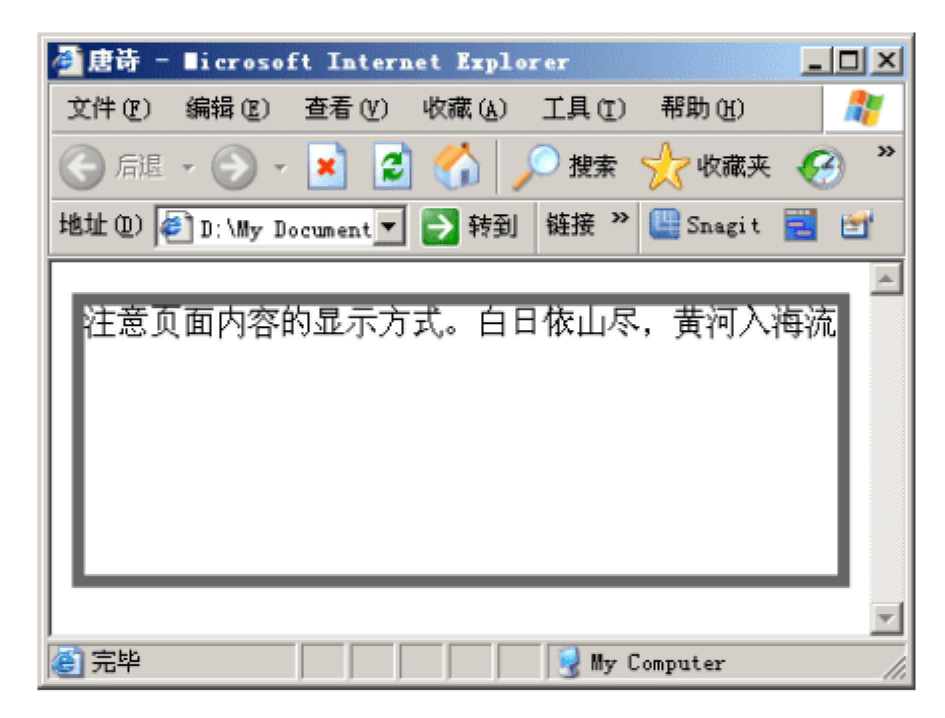


图 4-17 使用<nobr>元素的显示效果



图 4-18 取消<nobr>元素的显示效果

<nobr>元素中，不含有任何属性。

4.4.5 引用元素<q>

引用元素<q>用来在内容中定义一段引用的文本。由于各个浏览器对该属性支持情况不同，在不同的浏览器中，可能会有不同的显示效果。语法结构如下所示。

```
<q>……</q>
```

下面是使用<q>元素的实例。其代码如下所示。

例程 4-16 q.html

```
<body>
在这个页面中，注意引用<q>大江东去浪淘尽</q>中内容的显示方式。
</body>
```

其代码运行后，在 IE 浏览器中的显示效果如图 4-19 所示。在 Firefox 浏览器中的显示效果如图 4-20 所示。

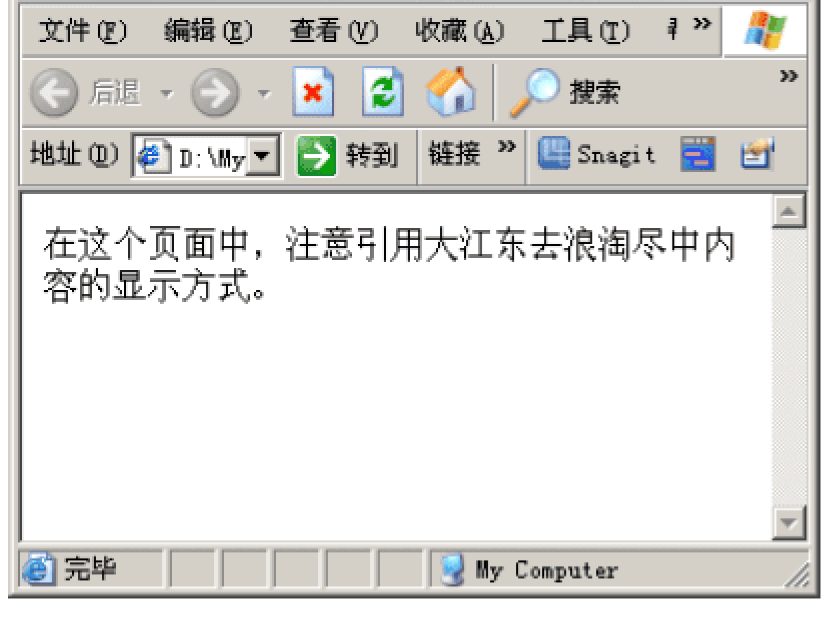





图 4-19　使用<q>元素在 IE 中的显示效果

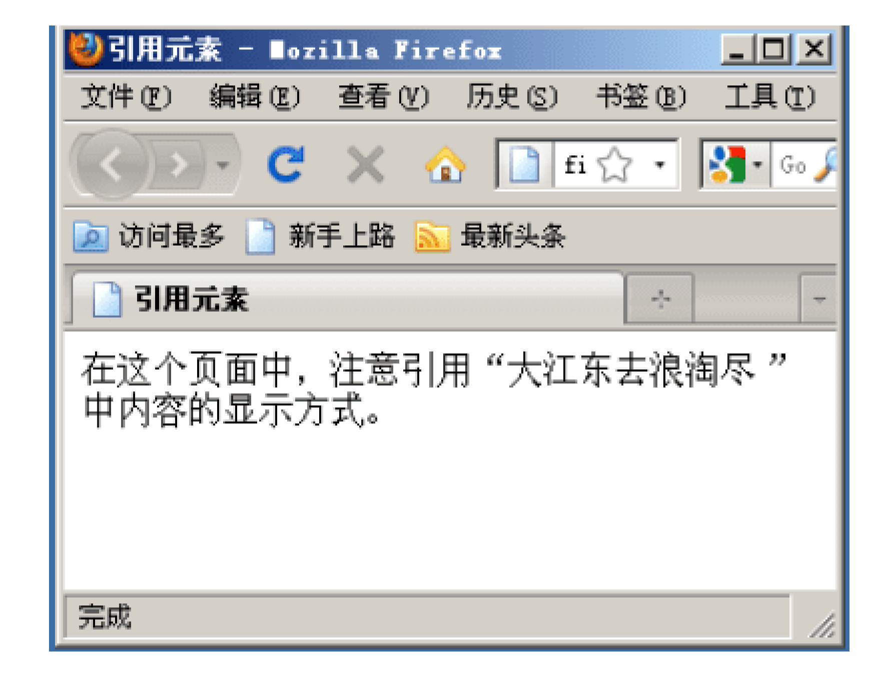


图 4-20　使用<q>元素在 Firefox 中的显示效果

从图 4-19 和图 4-20 可以看出，在 Firefox 浏览器中被<q>元素包含的文本，会显示在“引号”之中，而在 IE 浏览器中没有特殊的显示效果。<q>元素中可以使用的所有属性，如表 4-9 所示。

表 4-9　<q>元素的所有属性

属 性 名 称	写　法
文本显示方向属性	dir
指定语言属性	lang
类属性	class
定义级联样式属性	style
标记属性	id
标题属性	title
引用地址属性	cite

其中 cite 属性，用来指定<q>元素内容的引用地址，目前为止还没有任何效果。

4.4.6　地址元素<address>

地址元素<address>用来在内容中定义地址的相关内容。在常用的浏览器中，会将<address>元素所包含的内容用斜体显示。语法结构如下所示。

```
<address>……</address>
```

下面是使用<address>元素的实例。其代码如下所示。

例程 4-17　address.html

```
<body>
<address>Email:hao123@126.com<br />
 地址：北京市<br />
 电话：000000000</address>
</body>
```

02 行中使用了地址属性。其代码运行后，显示效果如图 4-21 所示。

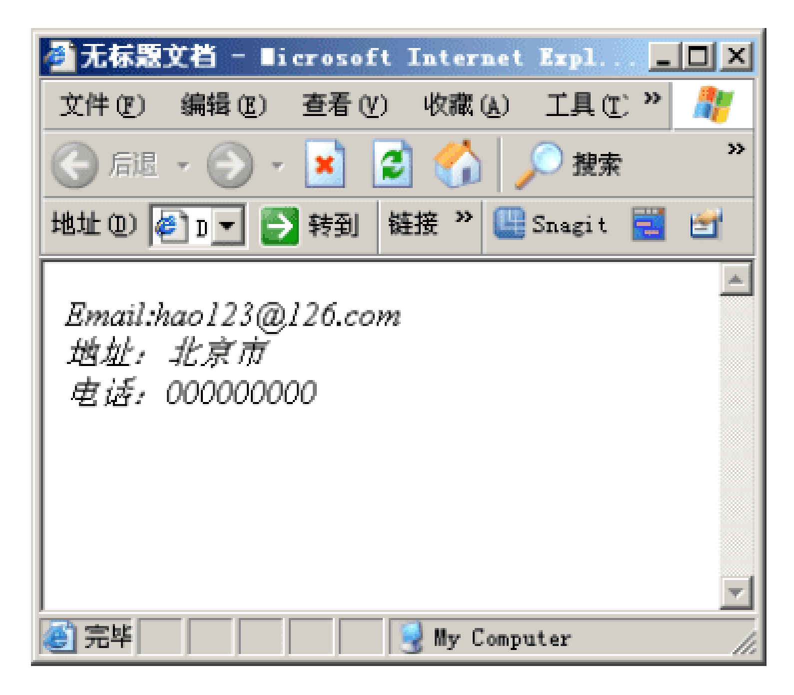


图 4-21　使用<address>元素的显示效果

<address>元素中可以使用的所有属性，如表 4-10 所示。

表 4-10　<address>元素的所有属性

代码写法	属性名称
dir	文本显示方向属性
lang	指定语言属性
class	类属性
style	定义级联样式属性
id	标记属性
title	标题属性

4.5 水平分隔线元素<hr>

水平分隔线元素<hr>用一条一定高度的分隔线，分隔页面内容。在使用<hr>元素的地方，文本将换行显示。语法结构如下所示。

```
<hr />
```

下面是使用<hr>元素的实例。其代码如下所示。

例程 4-18　hr.html

```
01  <body>
注意页面的显示方式，唐诗<hr />白日依山尽，黄河入海流。
02  </body>
```

其代码运行后，注意在</hr>的地方会出现水平分隔线，显示效果如图 4-22 所示。

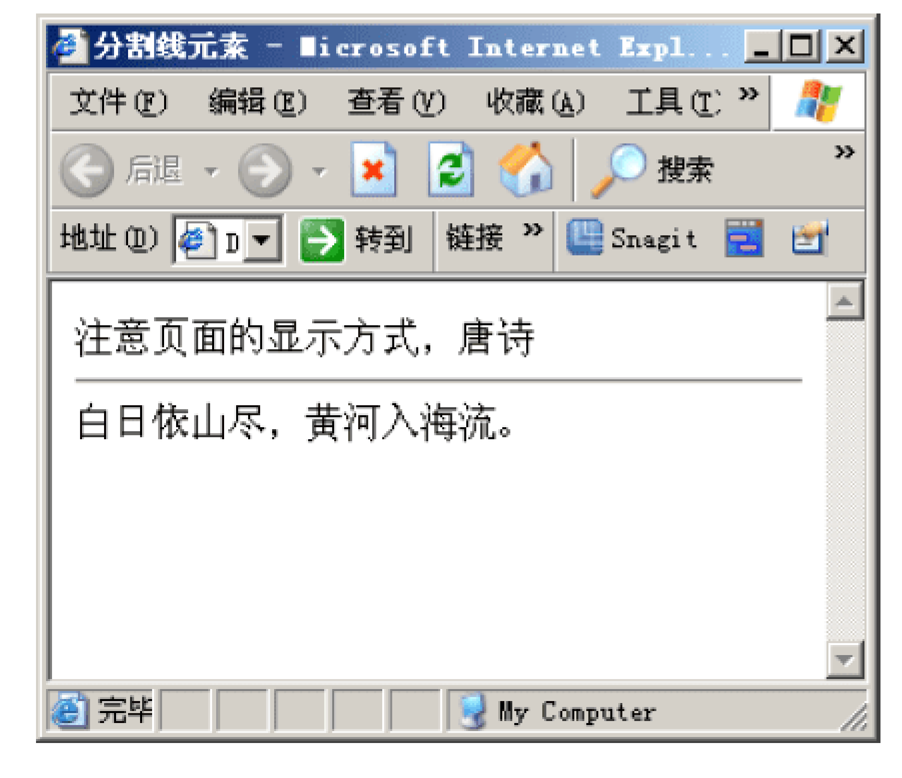


图 4-22　使用<hr>元素的显示效果

在没有定义任何属性的情况下，<hr>元素的宽度将和容器元素的宽度相同。<hr>元素中可以使用的所有属性，如表 4-11 所示。

表 4-11　<hr>元素的所有属性

代码写法	属性名称
dir	文本显示方向属性
lang	指定语言属性
class	类属性
style	定义级联样式属性
id	标记属性
title	标题属性
color	颜色属性
align	对齐属性
width	宽度属性
size	厚度属性

4.5.1　高度属性 size

水平分隔线的厚度属性 size，用来定义水平分隔线的粗细。水平分隔线的显示效果是一种凹陷的 3D 效果，使用 size 属性，将能够更改分隔线的厚度。语法结构如下所示。

```
<hr size="数字值" />
```

数字值的单位是像素。下面是使用 size 属性的实例。其代码如下所示。

例程 4-19　hr-size.html

```
<body>
注意页面的显示方式，中间使用<hr size="30" />空山新雨后，天气晚来秋。
</body>
```

02 行中定义了水平线的高度，其代码运行后，页面中间会出现高度为 30 的分隔线，显示效果如图 4-23 所示。

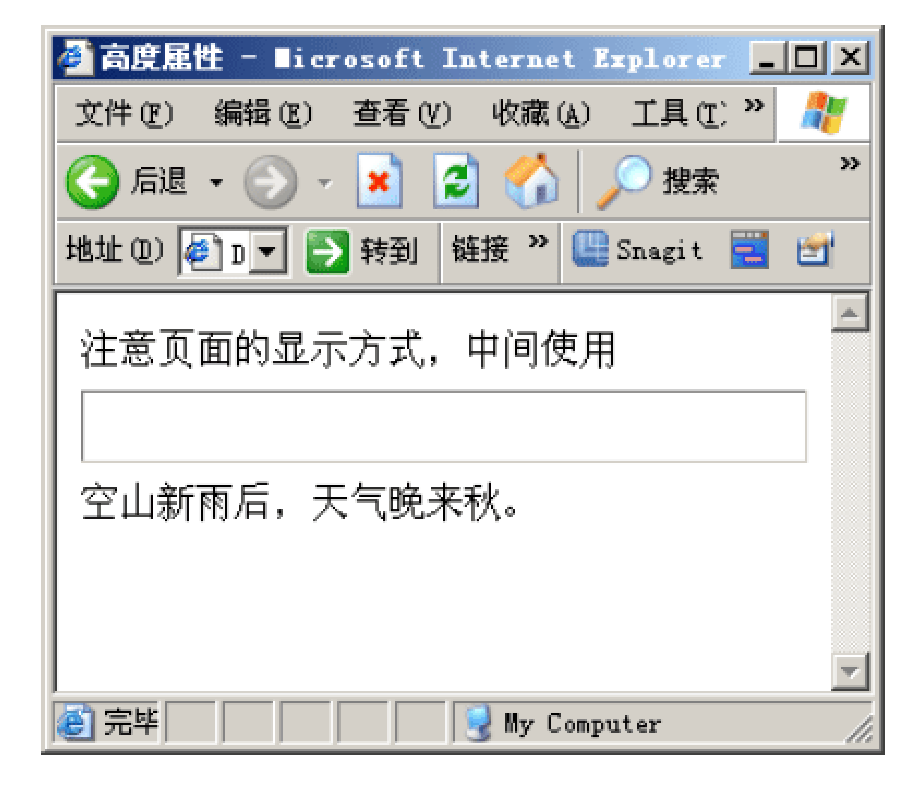


图 4-23　使用 size 属性的显示效果

4.5.2　样式属性 noshade

水平分隔线的样式属性 noshade 用来更改水平分隔线默认的 3D 效果为平面的 2D 效果。同时水平分隔线将会以更改后的格式显示。语法结构如下所示。

```
<hr noshade =" noshade" />
```

noshade 属性中没有任何取值，所以要用自身名称作为值。

下面是使用 noshade 属性的实例。其代码如下所示。

例程 4-20　hr-noshade.html

```
01  <body>
没使用 noshade 属性的分隔线
02  <hr size="20" />
使用 noshade 属性的分隔线
03  <hr size="20" noshade="noshade" />
04  </body>
```

说明

为了明显看出使用属性后的显示效果，用一个没有使用 noshade 属性的水平分隔线作为对比。

其代码运行后，在 IE 浏览器中的显示效果，如图 4-24 所示。在 Firefox 浏览器中的显示效果，如图 4-25 所示。



图 4-24　使用 size 属性在 IE 中的显示效果

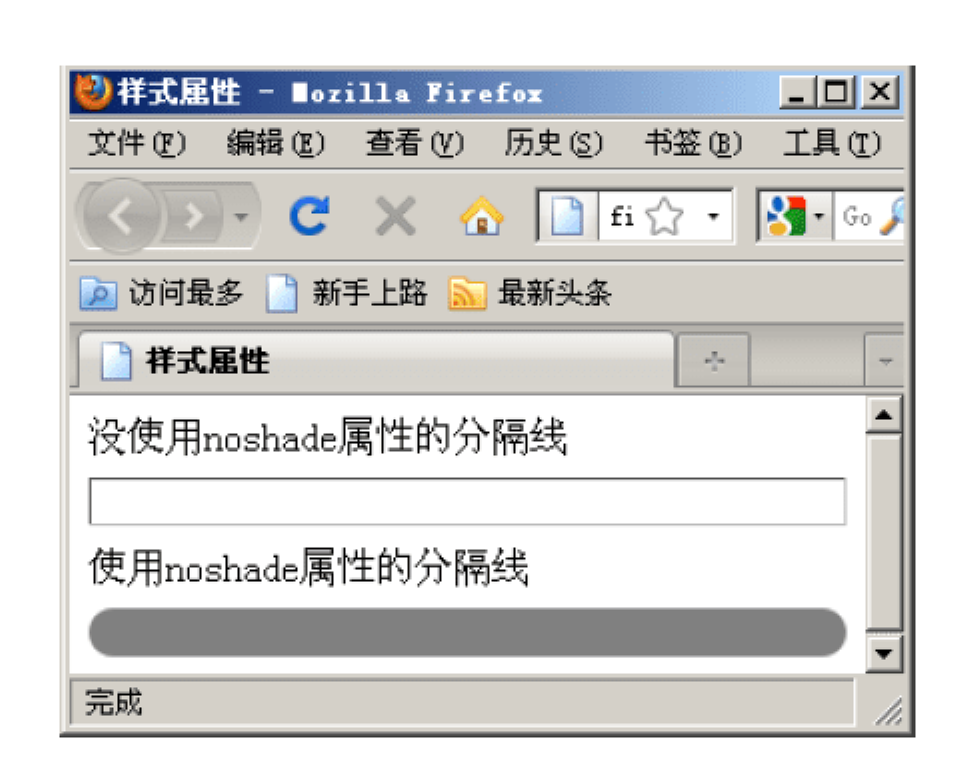


图 4-25　使用 size 属性在 Firefox 中的显示效果

4.5.3 宽度属性 width

水平分隔线的宽度属性 width 用来定义水平分隔线的宽度。语法结构如下所示。

```
<hr width="数字值" />
```

数字值的单位是像素。下面是使用 width 属性的实例。其代码如下所示。

例程 4-21 hr-width.html

```
01 <body>
注意页面的显示特点，在水平分隔线中使用 width 属性的实例
02 <hr width="200" />
03 </body>
```

02 行中使用了宽度属性 width，其代码运行后，显示效果如图 4-26 所示。

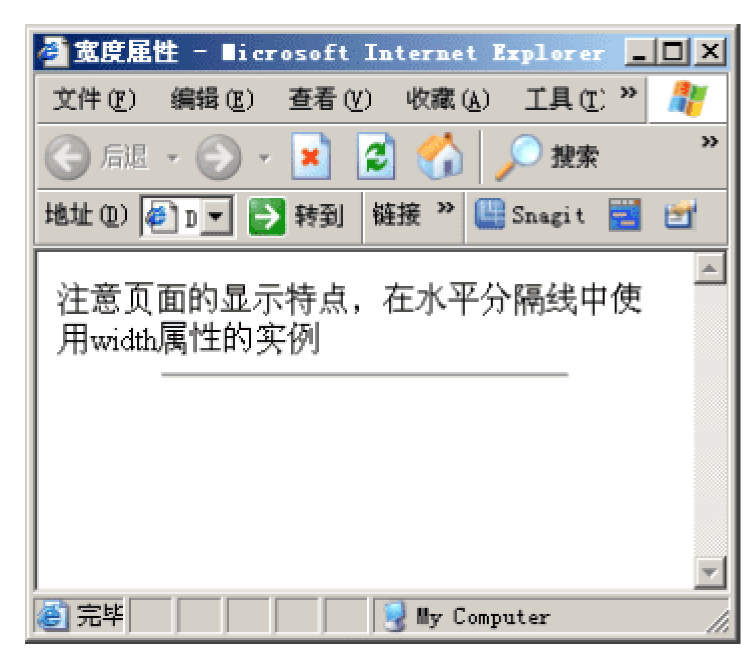


图 4-26 使用 width 属性的显示效果

4.5.4 对齐属性 align

水平分隔线的对齐属性 align 用来定义水平分隔线的对齐方式。与<div>元素的 align 属性有所区别，因为<hr>元素不能包含内容，所以 align 属性是指元素本身相对于父元素的对齐方式。语法如下所示。

```
<hr align="属性值" />
```

align 属性的具体取值和含义如表 4-12 所示。下面是使用 align 属性的实例。其代码如下所示。

表 4-13 align 属性的取值及含义

属性代码	代码含义
center	居中对齐(默认值)
left	左侧对齐
right	右侧对齐

例程 4-22 hr-align.html

```
<body>
水平分隔线用左对齐
<hr width="300" align="left" />
水平分隔线用中间对齐
<hr width="300" align="center" />
水平分隔线用右对齐
<hr width="300" align="right" />
</body>
```

02 到 04 行中都设置水平线的长度为 300，分别定义了 3 条水平线以左对齐、中间对齐和右对齐显示，其代码运行后的显示效果如图 4-27 所示。

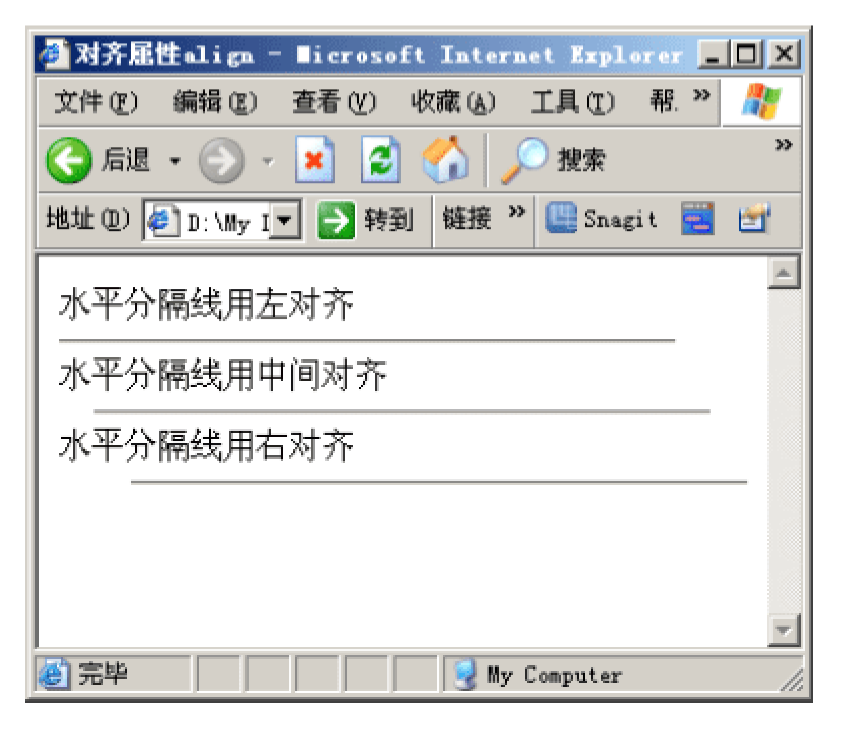


图 4-27　使用 align 属性的显示效果

4.5.5　颜色属性 color

水平分隔线的颜色属性 color 用来定义水平分隔线的显示颜色。3D 形式的水平分隔线默认是没有颜色的，2D 形式的水平分隔线默认的颜色是灰色。语法结构如下所示。

```
<hr color="颜色值" />
```

下面是使用 color 属性的实例。其代码如下所示。

例程 4-23　hr-color.html

```
01    <title>颜色属性</title>
02    <style type="text/css">
<!--
.STYLE1 {color: #0000FF}
-->
03    </style>
04    </head>
05    <body>
06    <span class="STYLE1"> 水平分隔线中使用 color 属性</span>
07    <hr size="20" color="#999999" />
08    </body>
```

在本例程中，02 和 03 行中设置字体的颜色为蓝色，07 行中设置水平线的宽度为 20 像素，颜色为浅灰色，位置为默认属性，其代码运行后的显示效果如图 4-28 所示。

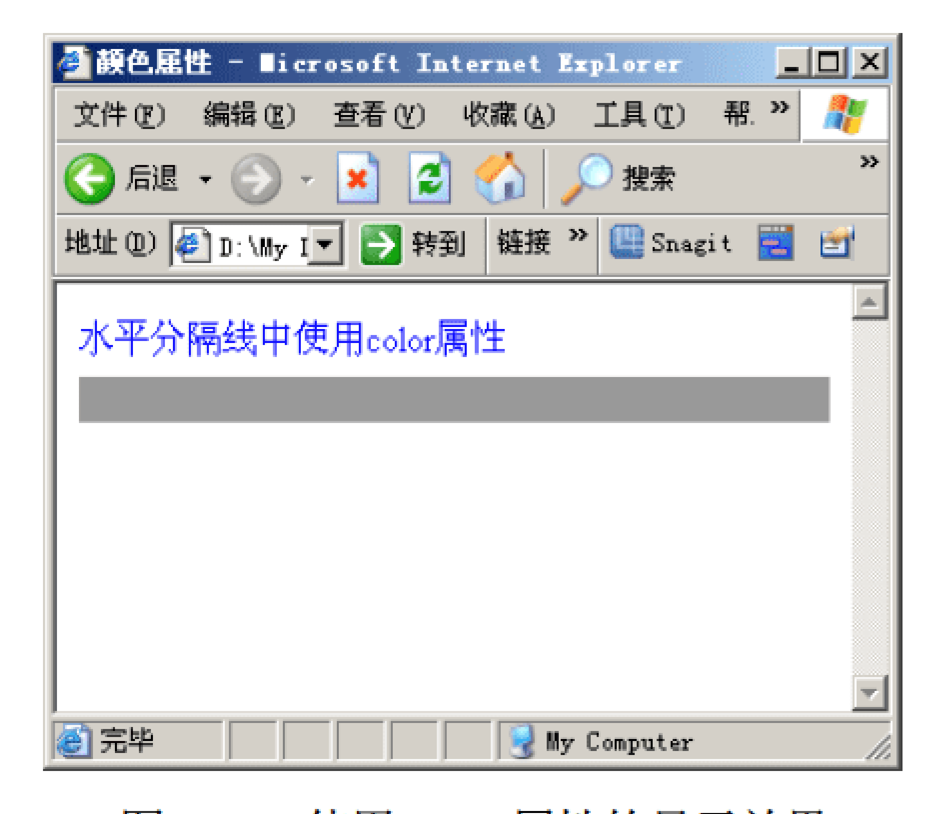


图 4-28　使用 color 属性的显示效果

4.6 基于物理样式的文本元素

基于物理样式的文本元素用来显示文本的加粗、等宽、上下标、下划线等样式。其中包括<b>、<i>、<sub>等元素。其中有些元素的表现效果可能相同，也有一些元素浏览器并不会做任何处理。基于物理样式的文本元素中，可以使用的所有属性如表 4-13 所示。

表 4-13　基于物理样式的文本元素的所有属性

代 码 写 法	属 性 名 称
dir	文本显示方向属性
lang	指定语言属性
class	类属性
style	定义级联样式属性
id	标记属性
title	标题属性

4.6.1　加粗元素<b>

加粗元素<b>用来使包含的文本加粗显示。语法结构如下所示。

```
<b>……</b>
```

下面是使用<b>元素的实例。其代码如下所示。

例程 4-24　b.html

```
01　<body>
02　<b>加粗的字体显示</b>普通文字显示。
03　</body>
```

注意 02 行中<b>元素之间内容的显示方式，其代码运行后的显示效果如图 4-29 所示。



图 4-29　使用<b>元素的显示效果

4.6.2　放大元素<big>

放大元素<big>用来使包含的文本增大一号显示。当文本内容已经是最大字号时，将不能继续增大。语法结构如下所示。

```
<big>……</big>
```

下面是使用<big>元素的实例。其代码如下所示。

例程 4-25　big.html

```
<body>
<big>放大的字体的显示方式</big>普通文字显示方式。
</body>
```

其代码运行后，显示效果如图 4-30 所示。



注意

可以在<big>元素中再次使用<big>元素，其达到的效果是文本再次放大一号。

下面是<big>元素中再次使用<big>元素的实例。其代码如下所示。

例程 4-26　bigtwo.html

```
<body>
<big><big>放大的字体的显示方式</big></big>普通文字显示方式。
</body>
```

其代码运行后，显示效果如图 4-31 所示。注意图 4-30 与图 4-31 的效果差别。



图 4-30　使用<big>元素的显示效果

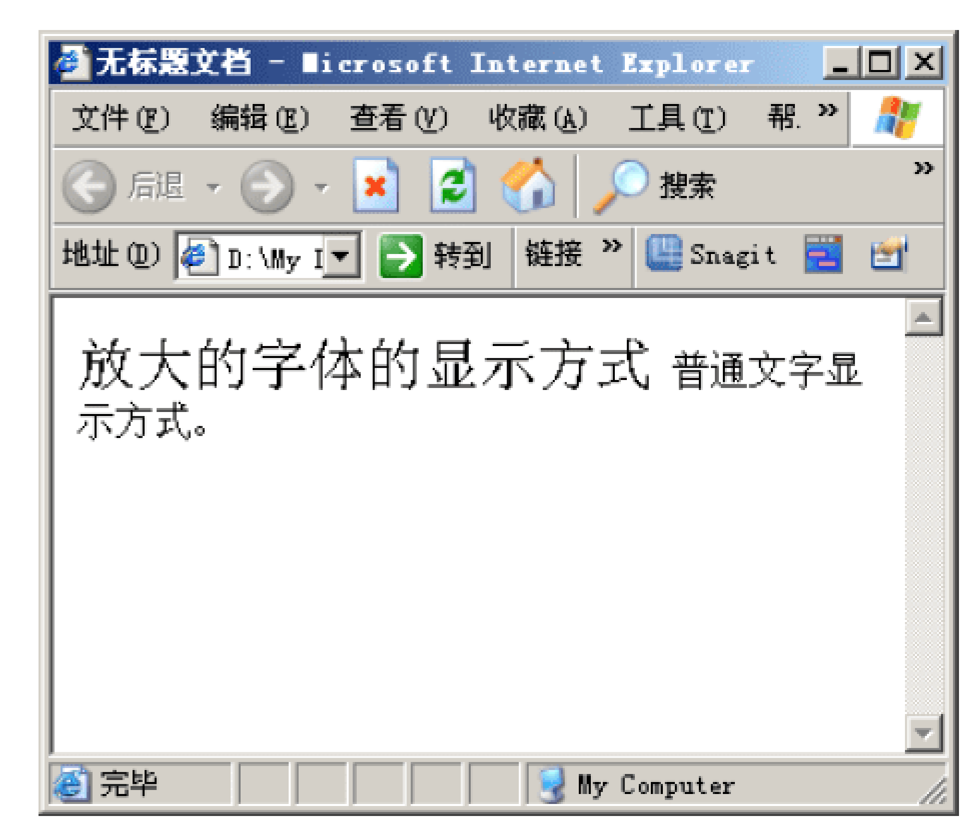


图 4-31　嵌套的<big>元素的显示效果

4.6.3　缩小元素<small>

缩小元素<small>，用来使包含的文本缩小一号显示。当文本内容已经是最小字号时，将不能继续变小。语法结构如下所示。

```
<small>……</small>
```

下面是使用<small>元素的实例。其代码如下所示。

例程 4-27　small.html

```
<body>
<small>缩小的字体</small>普通字体。
```

```
03    </body>
```

02 行中<small>元素之间的内容会缩小。其代码运行后，显示效果如图 4-32 所示。同样，<small>元素也可以嵌套使用。其代码如下所示。

例程 4-28　small.html

```
01    <body>
02    <small><small>缩小的字体</small>
      </small>
普通文本。
03    </body>
```

02 行中嵌套使用缩小元素，这些文本会以更小的形式显示。其代码运行后，显示效果如图 4-33 所示。



图 4-32　使用<small>元素的显示效果



图 4-33　嵌套的<small>元素的显示效果

4.6.4　斜体显示元素<i>

斜体显示元素<i>，用来使包含的文本内容以斜体的方式显示。语法结构如下所示。

```
<i>……</i>
```

下面是使用<i>元素的实例。其代码如下所示。

例程 4-29　i.html

```
01    <body>
02    <i>斜体显示的文本</i>普通文本。
03    </body>
```

02 行中使用了斜体显示元素<i>，其代码运行后，显示效果如图 4-34 所示。



图 4-34　使用<i>元素的显示效果

4.6.5　下标元素<sub>

下标元素<sub>用来使包含的文本内容以下标的方式

显示。<sub>元素中的文本的顶部，将在普通文本的一半高度上显示。语法结构如下所示。

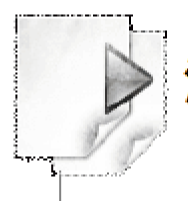
```
<sub>……</sub>
```

下面是使用<sub>元素的实例。其代码如下所示。

例程 4-30 sub.html

```
<body>
唐诗无题<sub>作者李商隐</sub>身无彩凤双飞翼，心有灵犀一点通。
</body>
```

<sub>中的文字就是下标部分，其代码运行后，显示效果如图 4-35 所示。



注意

连续使用两个下标元素并不会产生叠加效果

下面是连续使用两个下标元素的实例。其代码如下所示。

例程 4-31 subtwo.html

```
<body>
唐诗无题<sub>作者李商隐</sub><sub>是一首优秀的七绝</sub>身无彩凤双飞翼，心有灵犀一点通。
</body>
```

其代码运行后，显示效果如图 4-36 所示。

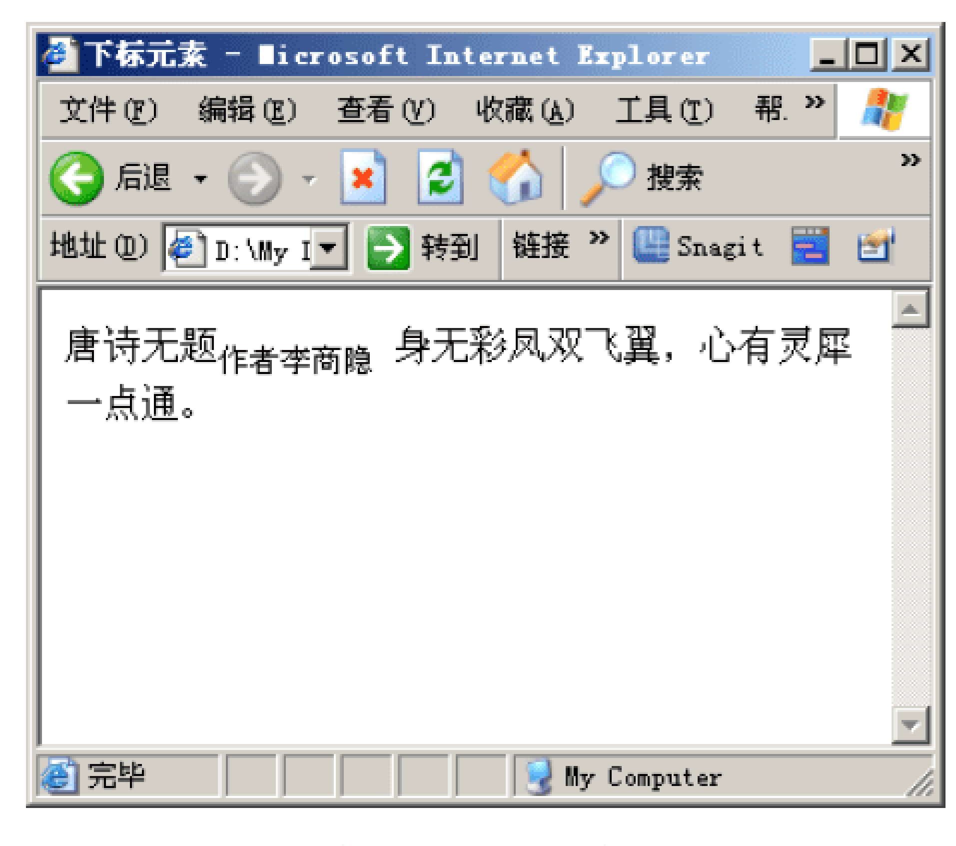


图 4-35 使用<sub>元素的显示效果

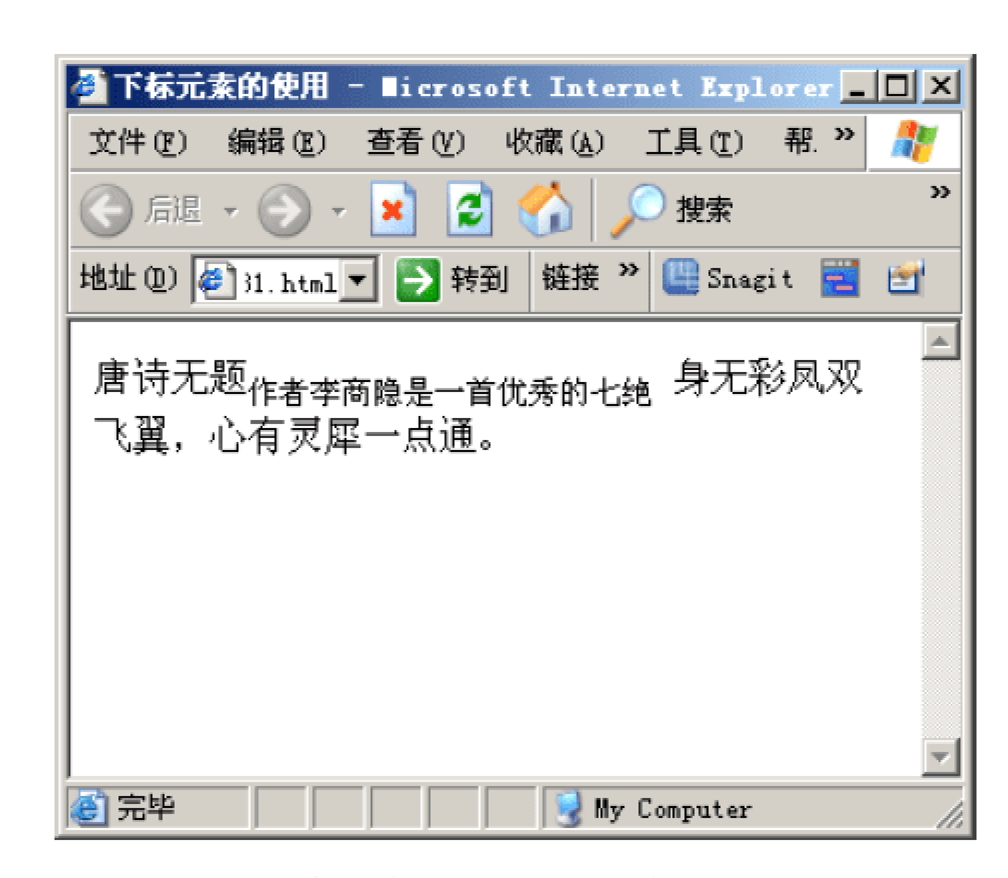


图 4-36 连续使用<sub>元素的显示效果

嵌套使用<sub>元素可以产生叠加的效果。下面是使用嵌套的两个下标元素的实例。代码如下所示。

例程 4-32 subthree.html

```
<body>
普通文本<sub>下标中显示的文本<sub>注意这里面下标显示的文本</sub></sub>普通文本。。
</body>
```

其代码运行后，显示效果如图 4-37 所示。

可以看到嵌套的两个下标元素后，页面中会在下标文字中再次出现下标。

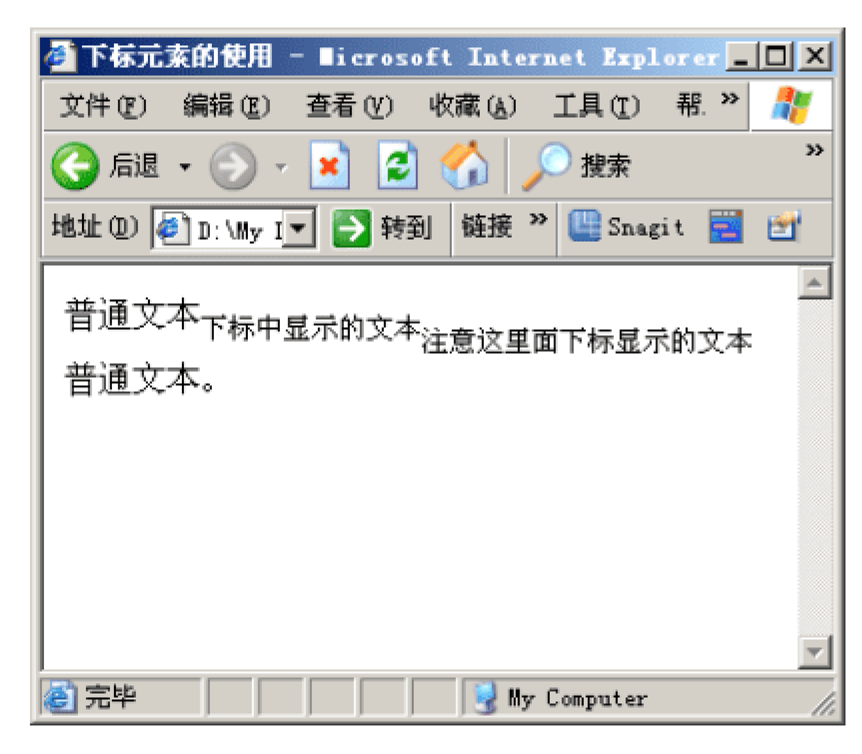
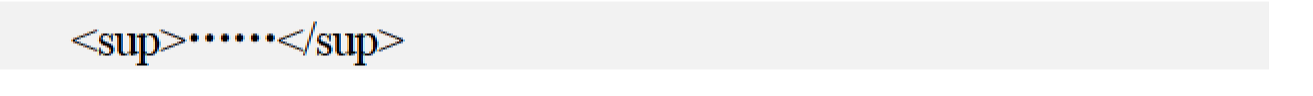


图 4-37　嵌套使用<sub>元素的显示效果

4.6.6　上标元素<sup>

上标元素<sup>用来使包含的文本内容以上标的方式显示。<sup>元素中的文本的底部，将在普通文本的一半高度上显示。语法结构如下所示。



```
<sup>……</sup>
```

下面是使用<sup>元素的实例。其代码如下所示。

例程 4-33　sup.html

```
<body>
普通位置<sup>注意这些文字的位置</sub>
</body>
```

其代码运行后，<sup>元素之间的内容将显示在配图文本的上面。显示效果如图 4-38 所示。

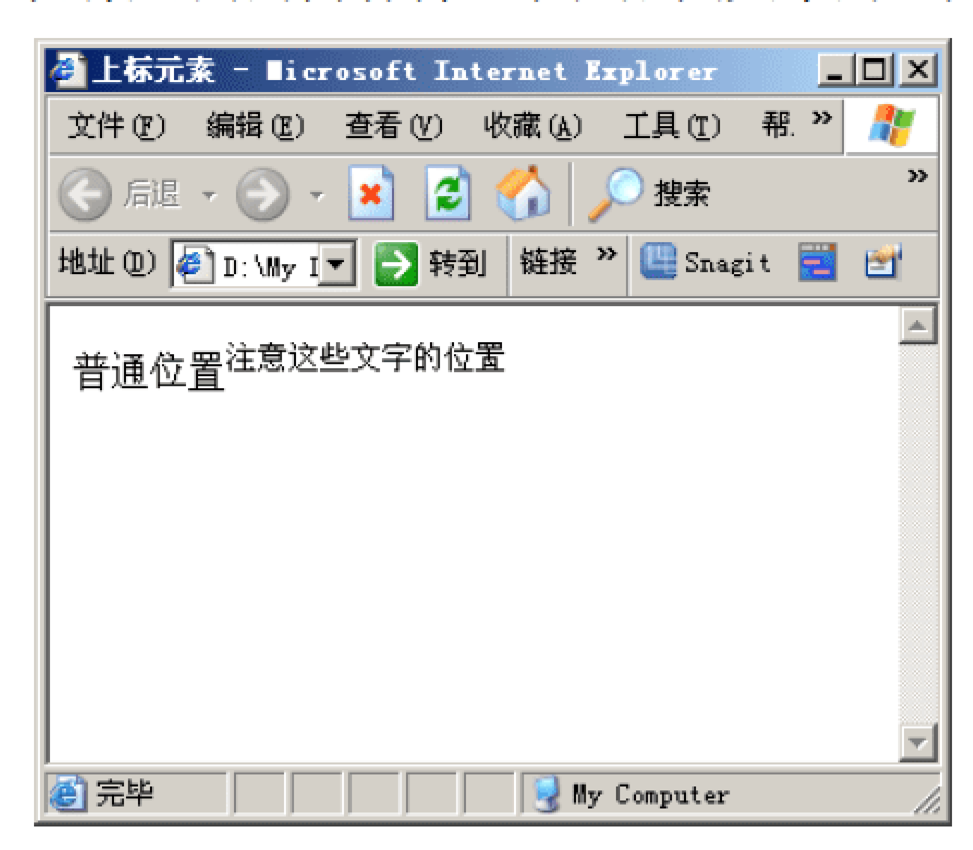


图 4-38　使用<sup>元素的显示效果

4.7　基于内容的文本元素

在 HTML 中，含有用来改变文本表现效果的元素。这些元素可以分为两类。一类是基于内容的文本元素，其含义是按照文本的功能性来定义元素。另一类是基于物理样式的文本元素，含义是用来改变文本的表现效果。基于内容的文本元素，有很好的语义性，包括<em>、<cite>、<strong>等元素。其中有一些元素可能有相同的表现效果，也有一些元素，浏览器并不会做任何处理。基于内容的文本元素中，可以使用的所有属性如表 4-14 所示。

表 4-14　基于内容的文本元素的所有属性

代 码 写 法	属 性 名 称
dir	文本显示方向属性
lang	指定语言属性
class	类属性
style	定义级联样式属性
id	标记属性
title	标题属性

4.7.1　强调元素<em>

强调元素<em>用来强调内容。一般被强调的部分会以斜体显示，用于和普通文本相区别。语法结构如下所示。

```
<em>……</em>
```

下面是使用<em>元素的实例。其代码如下所示。

例程 4-34　em.html

```
01  <body>
02  <em>注意这些文字的显示</em>普通页面文文字
03  </body>
```

02 行中代码<em>中间的部分会以斜体的形式出现，运行代码后，显示效果如图 4-39 所示。

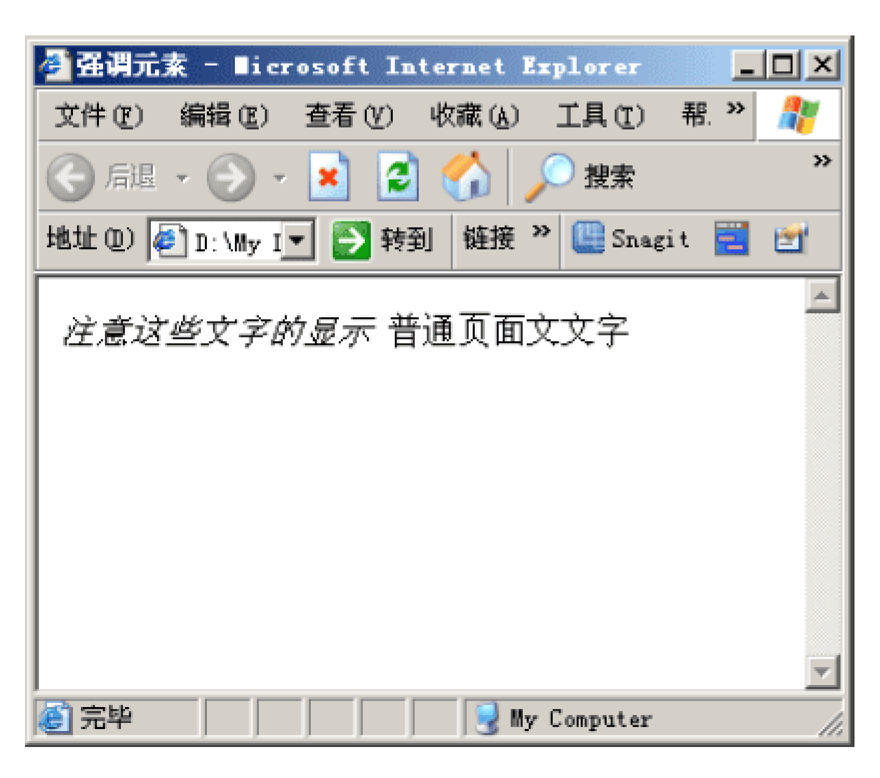


图 4-39　使用<em>元素的显示效果

4.7.2　加粗的强调元素<strong>

加粗的强调元素<strong>用来强调内容。与<em>元素不同的是，使用<strong>元素的文本将会以加粗的格式显示，用于和普通文本相区别。语法结构如下所示。

```
<strong>……</strong>
```

下面是使用<strong>元素的实例。其代码如下所示。

例程 4-35　strong.html

```
<body>
<strong>这里是被强调的文本</strong>这里是普通文本
</body>
```

02 行中<strong>元素中间的部分会以加粗的形式显示。其代码运行后，显示效果如图 4-40 所示。

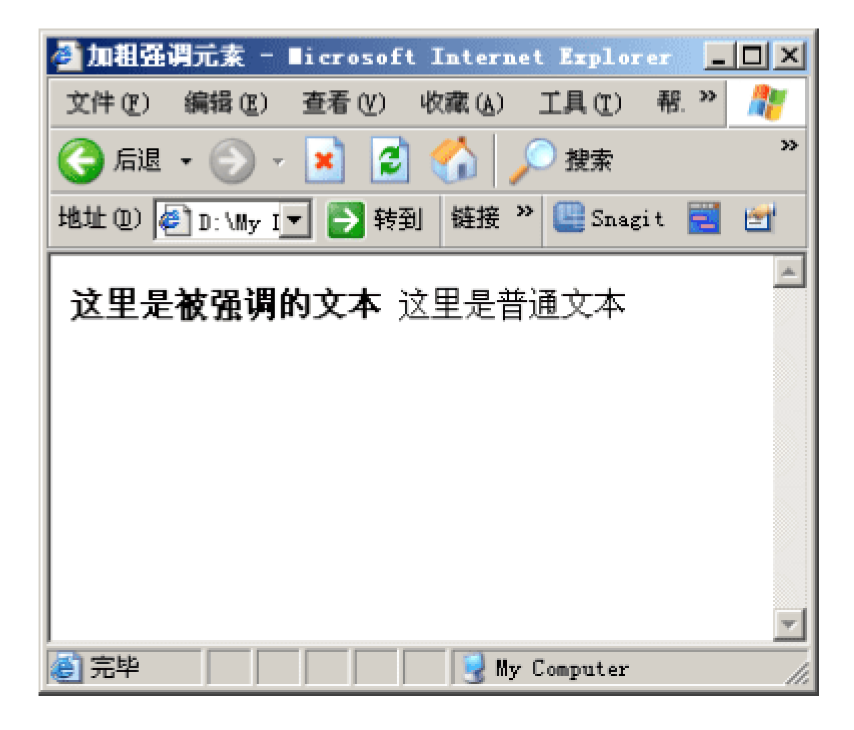


图 4-40　使用<strong>元素的显示效果

4.7.3　提取元素<samp>

提取元素<samp>从文本中提取部分内容，提取部分内容将会比其他内容小一些。语法结构如下所示。

```
<samp>包含的内容部分</samp>
```

下面是使用<samp>元素的实例。其代码如下所示。

例程 4-36　samp.html

```
01　<body>
02　<samp>这里被提取的文本。唐诗三百首</samp>这里是普通页面文本。唐诗三百
03　</body>
```

02 行中使用提取元素<samp>，其代码运行后，显示效果如图 4-41 所示。

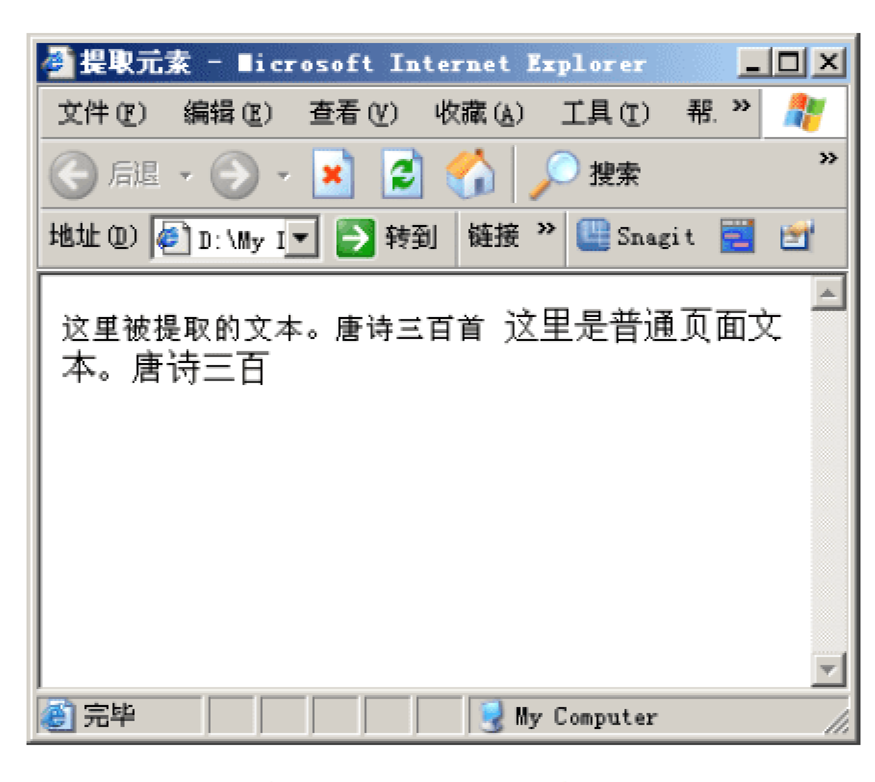


图 4-41　使用<samp>元素的显示效果

4.7.4　首字母缩写元素<acronym>

首字母缩写元素<acronym>用来显示包含内容的首字母缩写显示。由于浏览器到目前为止还不支持此属性，所以没有特殊的显示效果。语法结构如下所示。

```
<acronym>包含的内容部分</acronym>
```

由于浏览器到目前为止还不支持此属性，所以将不做实例演示了。

4.7.5　变量显示元素<var>

变量显示元素<var>用来显示源程序中的变量。和<code>元素类似，一般用来显示程序内

容。其包含的文本将会以斜体方式显示。语法结构如下所示。

```
<var>……</var>
```

下面是使用<var>元素的实例。其代码如下所示。

例程 4-37 var.html

```
01  <body>
02  <var>这里的部分是源程序的文本，</var>这里是普通页面文本
03  </body>
```

02 行中使用了变量显示元素<var>，其代码运行后，显示效果如图 4-42 所示。

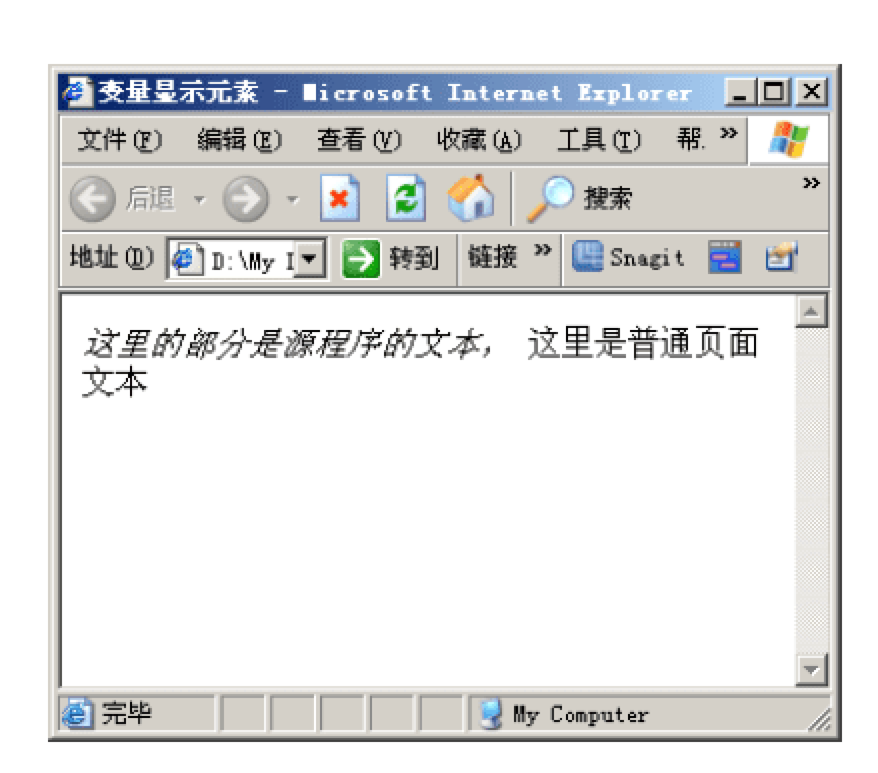


图 4-42 使用<var>元素的显示效果

4.7.6 文献参考元素<cite>

文献参考元素<cite>用来定义一段引用的文本内容。使用<cite>元素的文本，一般将会以斜体的格式显示，用来和普通文本相区别。语法结构如下所示。

```
<cite>……</cite>
```

下面是使用<cite>元素的实例。其代码如下所示。

例程 4-38 cite.html

```
01  <body>
02  <cite>这里是被引用的文本</cite>这里是普通文本
03  </body>
```

02 行中使用了文献参考元素<cite>。其代码运行后，<cite>元素之间内容将以斜体形式显示，显示效果如图 4-43 所示。



图 4-43 使用<cite>元素的显示效果

4.8 本章习题

一、选择题

1．在 HTML 中，标记<pre>的作用是(　)。

A. 标题标记　B．预排版标记　C. 转行标记　D．文字效果标记

2．在 HTML 中，具有对文字加粗作用的是(　)。

A. <b>　B．<big>　C. <em>　D．<strong>

3．网页中“换行符”对应的标签是(　)。

A. hr　B. div　C. br　D. p

4．层元素中用于设置文本对齐属性的是(　)。

A. class　B．style　C．align　D. id

5. 关于文本对齐，源代码设置不正确的一项是：(　)。

A. 居中对齐：<div align="middle">…</div>

B. 居右对齐：<div align="right">…</div>

C. 居左对齐：<div align="left">…</div>

D. 两端对齐：<div align="justify">…</div>

6．要控制水平线的粗细，应使用以下属性：________。

A. color

B. width

C. size

D. height

7．以下说法中，正确的是：________。

A. P 标记符与 BR 标记符的作用一样

B. 多个 P 标记符可以产生多个空行

C. 多个 BR 标记符可以产生多个空行

D. P 标记符的结束标记符通常不可以省略

二、填空题

1．<hr width=50%>表示创建一条 ______的水平线。

2．要设置一条 1 像素粗的水平线，应使用的 HTML 语句是________。

3．要使文字同时显示为粗体和斜体，应使用语句__________。

4．要是页面中的文字不自动换行，应该使用__________属性，而用_____属性取消制动换行。

5．设置一个宽度为 30px，长度为 200px，颜色为紫色的水平线的代码是______________.

三、实战练习

1．制作页面，加入一段文字，并要求用宽度为 30px，长度为 200px，颜色为黑色的水平线将文字分开。

2．制作页面，要使一段文字同时显示为粗体和斜体，另一段文字用到上标和下标。

3．制作页面，使用<div>元素调用 css 样式设置页面的文本显示效果。

Chapter 5

列表元素

列表元素，是用来定义条目信息数据的主要方法。按照功能和显示效果不同，列表元素可以大致分为无序列表、有序列表和定义列表三类。页面中通常使用的列表元素包括<ul>、<ol>、<dl>、<li>等。

本章主要内容有:

◎ 无序列表，使用<ul>和<li>元素定义。
◎ 有序列表，使用<ol>和<li>元素定义。
◎ 定义列表，使用<dl>、<dt>和<dd>元素定义。

5.1 无序列表元素

无序列表元素<ul>用来定义没有顺序编号的列表元素。例如，在网页的某个栏目中展示详细分类等。语法结构如下所示。

```
<ul>
<li>第 1 项
<li>第 2 项
<li>第 3 项
……
</ul>
```

下面是一个使用<ul>元素的实例。其代码如下所示。

例程 5-1 ul.html

```
01  <body>
02  <font size=5 color="#0033FF">提供下载的书籍：</font><br><br>
03  <ul>
     <li>历史故事
     <li>各类工具书
     <li>计算机教程
     <li>人物传记
04  </ul>
05  </body>
```

该实例中，03 和 04 行之间定义了一个含有 4 个条目的无序列表。其运行后的显示效果，如图 5-1 所示。

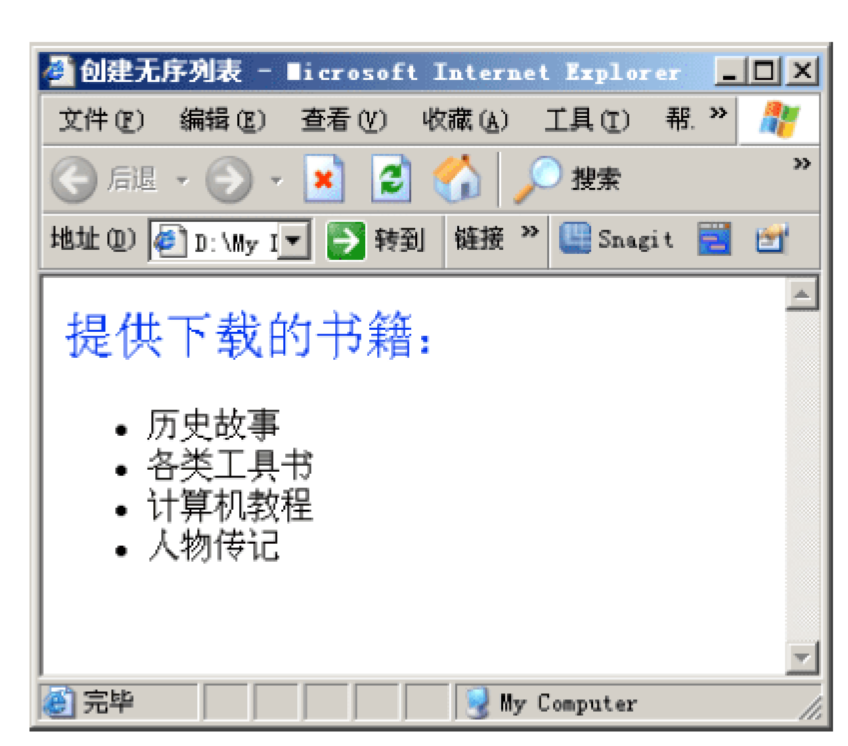


图 5-1　使用<ul>元素的显示效果

在<ul>元素中，不但可以使用文本内容，同时也可以使用图片内容。下面是在<ul>元素中使用图像元素的实例，其代码如下所示。

例程 5-2 ultwo.html

```
01  <body>
```

```
02  <ul>
  <li><img src="http://www.w3.org/Icons/right" alt="pic" />图片 1
    <li><img src="http://www.w3.org/Icons/right" alt="pic" />图片 2
    <li><img src="http://www.w3.org/Icons/right" alt="pic" />图片 3
    <li><img src="http://www.w3.org/Icons/right" alt="pic" />图片 4
    <li><img src="http://www.w3.org/Icons/right" alt="pic" />图片 5
03  </ul>
04  </body>
```

在该实例中，02 和 03 行之间使用了 W3C 官方站点的一个图像文件，其运行后的显示效果如图 5-2 所示。

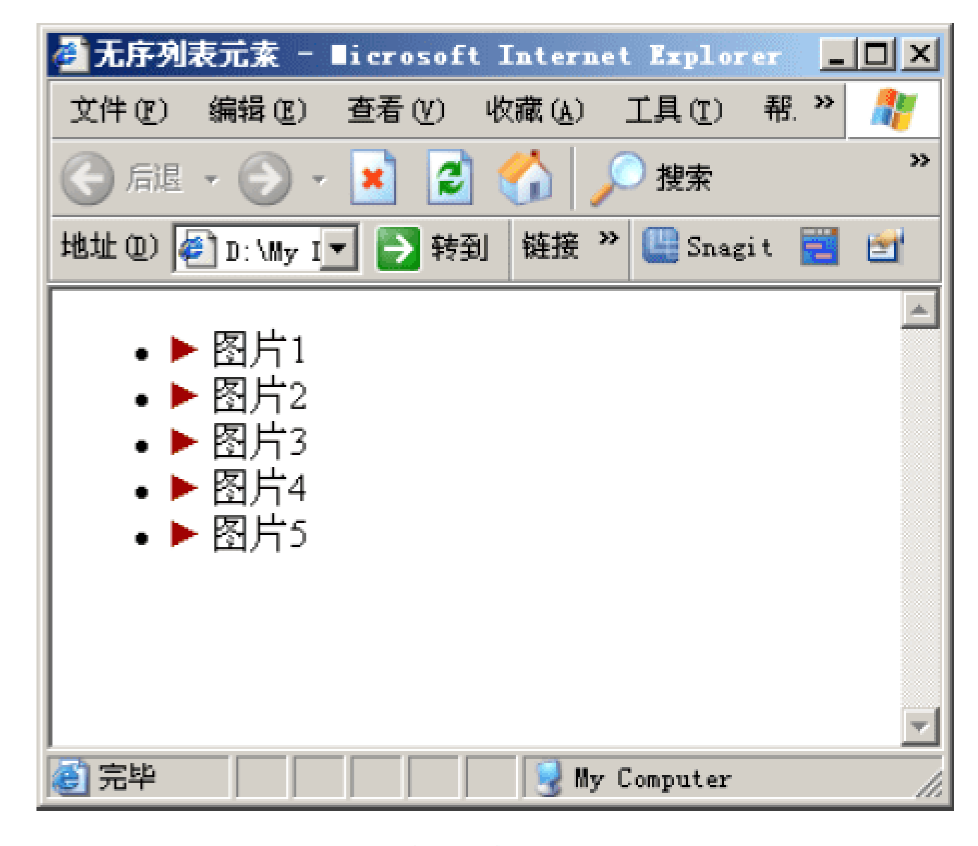


图 5-2　元素中使用图片的显示效果

从图 5-2 可以看出，当在列表中使用了图片元素后，列表前面的修饰部分，将默认地显示在列表行的底部。<ul>元素中可以使用的所有属性如表 5-1 所示。

表 5-1　元素的所有属性

属 性 名 称	写　法
文本显示方向属性	dir
指定语言属性	lang
类属性	class
定义级联样式属性	style
项目符号的类型属性	type
标题属性	title
标记属性	id

项目符号的类型属性 type 用来指定列表条目前的项目符号(列表条目前的修饰部分)的类型。语法结构如下所示。

```
<ul type=符号类型>
<li>第 1 项
<li>第 2 项
```

```
<li>第 3 项
……
</ul>
```

其中 type 属性的取值有 disc、circle 和 square 这 3 种，下面是一个使用 type 属性的实例。其代码如下所示。

例程 5-3　ul-type.html

```
01  <body>
02  <font size=5 color="#990000">一楼主要出售的商品种类：</font><br><br>
03  <ul type="square">
      <li>生活用品
      <li>蔬菜肉类
      <li>衣服
04  </ul>
05  <hr color="#CC0000" size=2>
06  <font size=5 color="#990000">二楼主要出售的电子产品：</font><br><br>
07  <ul type="circle">
      <li>手机
      <li>数码相机
      <li>计算机
      <li>学习机
      <li>音响设备
08  </ul>
09  </body>
```

03 和 07 行中使用<ul type=" ">的方法分别定义了条目前面的符号类型为方块和圆，其代码运行后，显示效果如图 5-3 所示。

无序列表的类型定义也可以在<li>项中进行，其语法结构如下所示。

```
<li type==符号类型>
```

这样定义的结果是对单个项目进行定义，实例代码如下。

例程 5-4　ul.html

```
01  <body>
02  <font size=5 >出售的图书种类：</font><br><br>
03  <ul>
    <li type=disc >电子杂志类书籍
    <li type=circle>人物传记类书籍
    <li type=square >社会科学类书籍
04  </ul>
05  </body>
```

03 和 04 行之间的各个条目中使用了不同的无序列表的类型，代码运行后显示效果如图 5-4 所示。

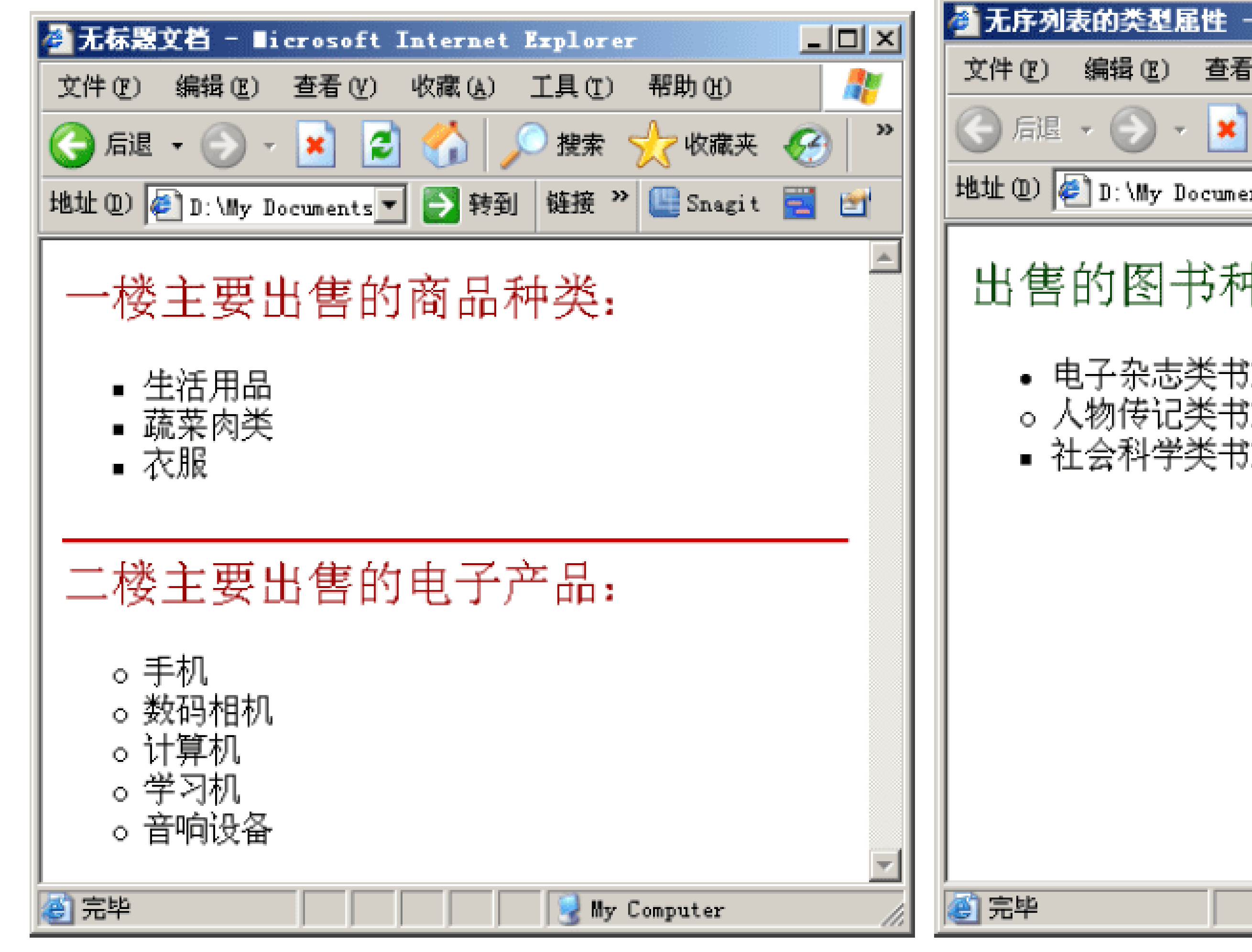


图 5-3 元素中使用 type 属性的显示效果

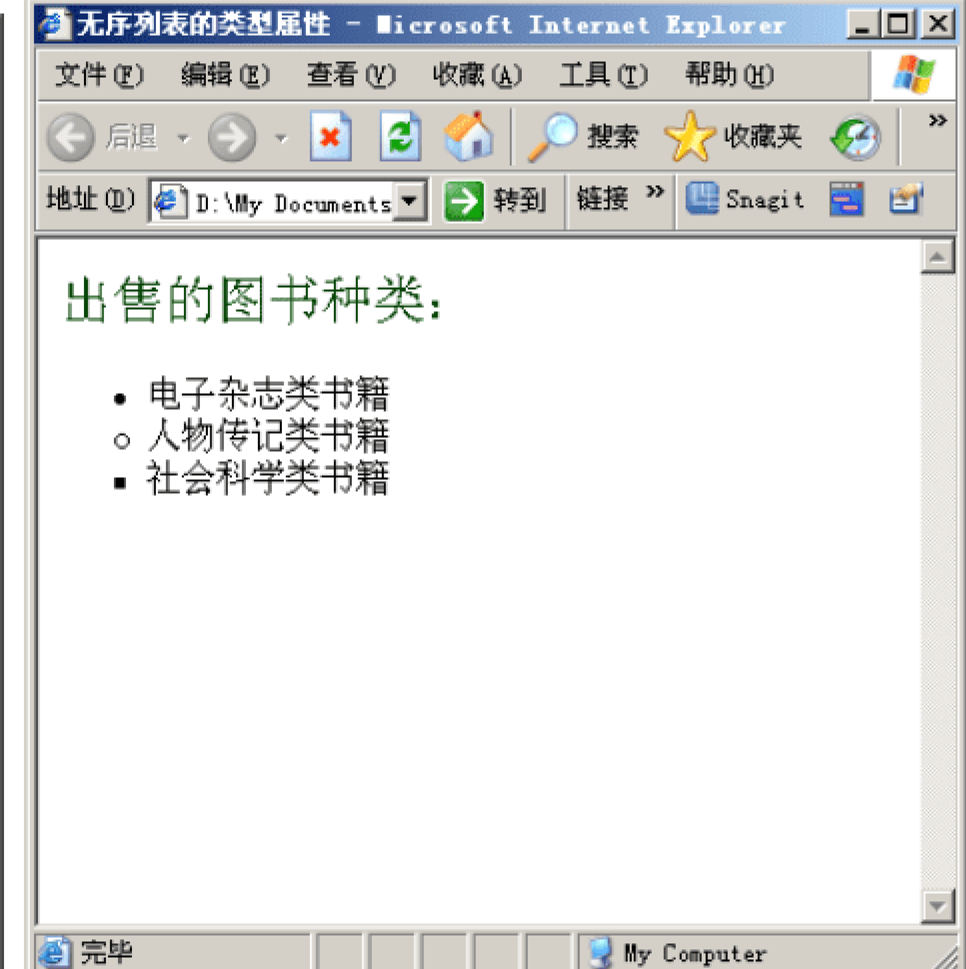
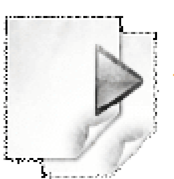


图 5-4 设置不同的项目符号

5.2 有序列表元素

有序列表元素<ol>，用来定义有顺序编号的列表元素。例如，一个教程的分步讲解等列表内容。语法结构如下所示。

```
<ol>
        <li>第 1 项
        <li>第 2 项
        <li>第 3 项
            ……
</ol>
```



注意

在网页中，<ol>元素一般也要和<li>元素一起使用。

下面是一个使用<ol>元素的实例。其代码如下所示。

例程 5-5 ol.html

```
<body>
<font size=5 color="#990000">本次图书推荐</font><br><br>
<ol>
        <li>古典文学
        <li>计算机编程
        <li>人物传记
        <li>现代文学
```

```
04  </ol>
05  </body>
```

该实例中，03 和 04 行之间定义了一个含有 4 个条目的有序列表。其运行后的显示效果，如图 5-5 所示。同样在<ol>元素中，也可以使用图片内容。下面是在<ol>元素中使用图像元素的实例，其代码如下所示。

例程 5-6 oltwo.html

```
01  <body>
02  <font size=5 color="#0033FF">引用图片</font><br><br>
03  <ol>
      <li><img src="http://www.w3.org/Icons/right" alt="pic" />图片 1
      <li><img src="http://www.w3.org/Icons/right" alt="pic" />图片 2
      <li><img src="http://www.w3.org/Icons/right" alt="pic" />图片 3
      <li><img src="http://www.w3.org/Icons/right" alt="pic" />图片 4
      <li><img src="http://www.w3.org/Icons/right" alt="pic" />图片 5
04  </ol>
05  </body>
```

在该实例中，03 和 04 行之间的各个条目都使用了 W3C 官方站点的一个图像文件，其运行后的显示效果，如图 5-6 所示。

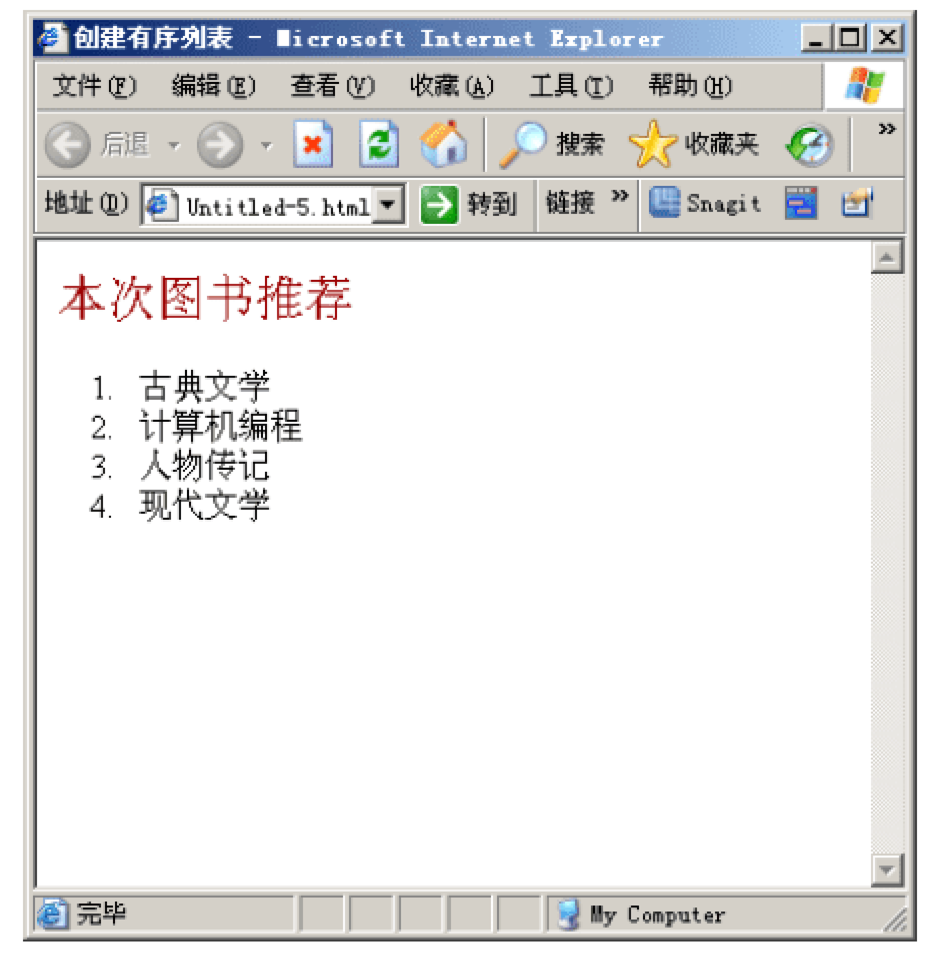


图 5-5　使用<ol>元素的显示效果



图 5-6　元素中使用图片的显示效果

从图 5-6 可以看出，在<ol>列表中，列表条目前的项目符号是从 1 开始依次增加的整数。同<ul>元素一样，项目符号将默认地显示在列表行的底部。

5.2.1　项目符号的类型属性 type

项目符号的类型属性 type 用来指定列表条目前的项目符号(列表条目前的修饰部分)的类型。

语法结构如下所示。

```
<ol type="属性值">其他元素</ol>
```

其中 type 属性的取值和含义，如表 5-2 所示。

表 5-2 type 属性的取值及含义

属 性 值	含 义
1	数字 1，2，3，4……
a	小写英文字母 a，b，c，d……
A	大写英文字母 A，B，C，D……
I	小写罗马数字 i，ii，iii，iv……
i	大写罗马数字 I，II，III，IV……

下面是一个使用 type 属性的实例。其代码如下所示。

例程 5-7 ol-type.html

```
01  <body>
02  <font size=4 color="#990000">音乐种类</font><br><br>
03  <ol type="A">
      <li>古典音乐
      <li>民间音乐
      <li>校园音乐
      <li>流行音乐
      <li>爵士音乐
04    </ol>
05  <hr size=2 color="#999999">
06  <font size=4 color="#990000">文化知识</font><br><br>
07  <ol type="I">
      <li>历史知识
      <li>地理知识
      <li>天文知识
      <li>政治知识
      <li>军事知识
      <li>经济知识
08  </ol>
09  </body>
```

代码中 03 行和 07 行中统一指定列表条目前的项目符号分别为大写英文字母 A 和罗马字母 I 开始，其代码运行后，显示效果如图 5-7 所示。

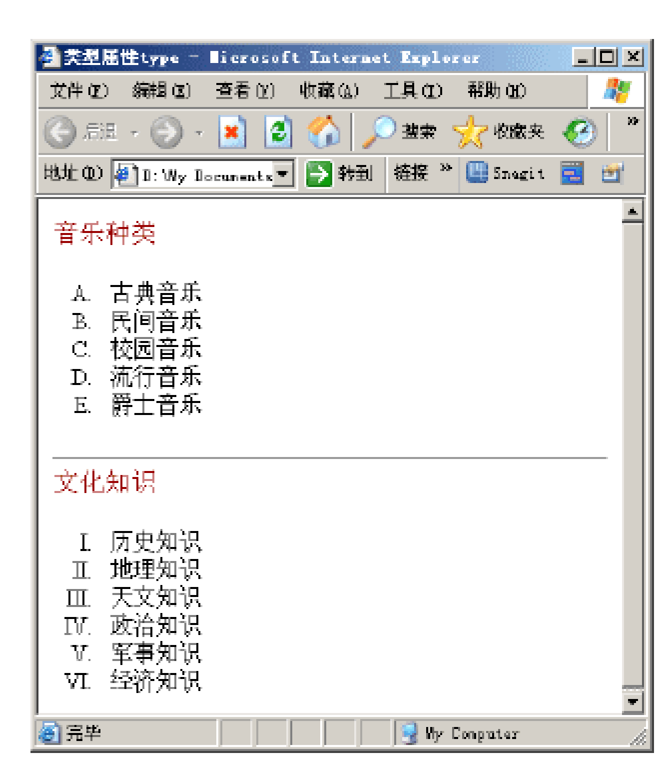


图 5-7 元素中使用 type 属性的显示效果

5.2.2 有序列表的起始值属性 start

有序列表的起始值属性 start 用来指定有序列表中项目符号的初始值。语法结构如下所示。

```
<ol start="数字值">其他元素</ol>
```

下面是一个使用 type 属性的实例。其代码如下所示。

例程 5-8　ol-start.html

```
01   <body>
02   <font size=4 color="#990000">音乐种类</font><br><br>
03   <ol type="A" start="6">
        <li>古典音乐
        <li>民间音乐
        <li>校园音乐
      <li>流行音乐
        <li>爵士音乐
04   </ol>
05   <hr size=2 color="#999999">
06   <font size=4 color="#990000">文化知识</font><br><br>
07   <ol type="I" start="5">
        <li>历史知识
        <li>地理知识
        <li>天文知识
        <li>政治知识
     <li>军事知识
     <li>经济知识
08   </ol>
09   </body>
```

代码中 03 行中指定列表条目前的项目符号从第 6 个大写英文字母开始，07 行中指定列表条目前的项目符号从第 5 个罗马字母开始。其代码运行后，显示效果如图 5-8 所示。

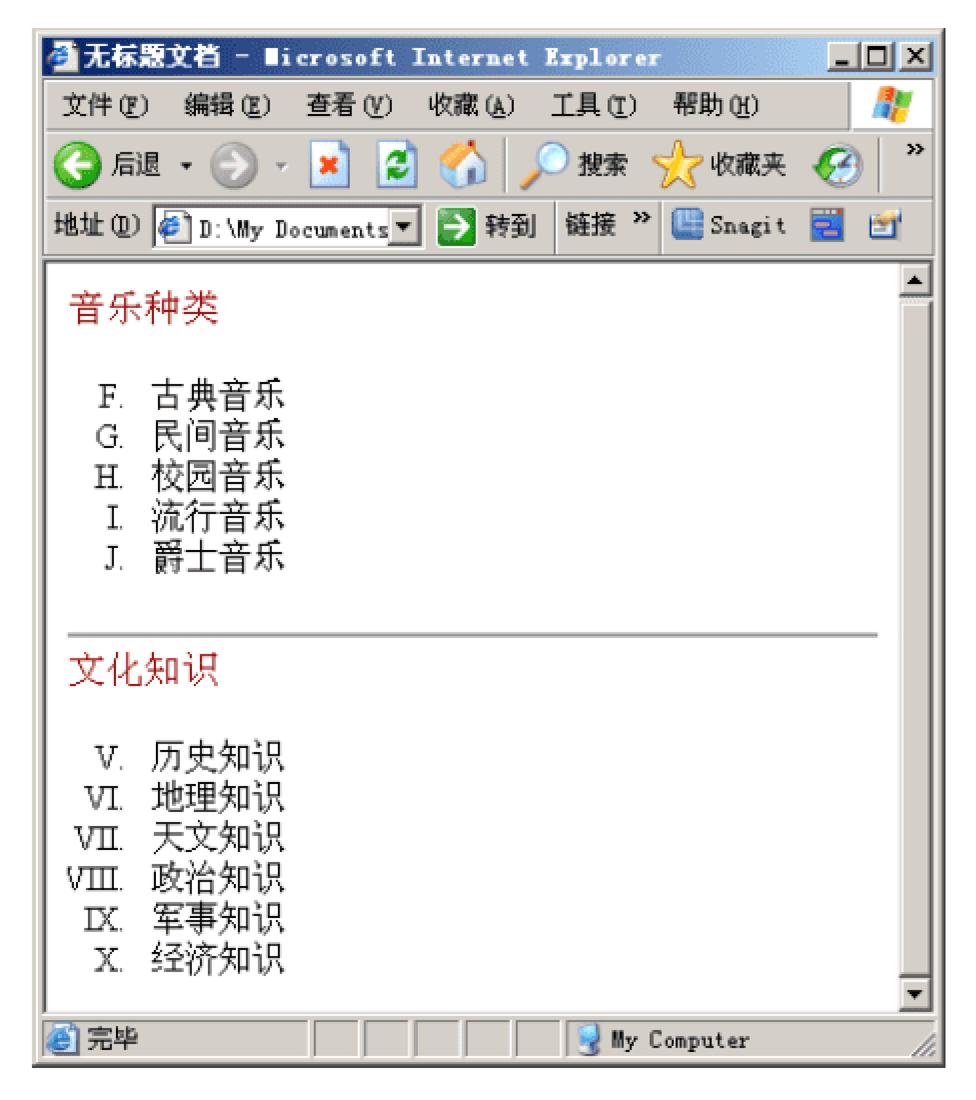


图 5-8　元素中使用 start 属性的显示效果

从图 5-8 可以看出，当定义了 start 属性后，第一个有序列表从第 6 个字母开始，第二个有序列表从大写罗马数字 5 开始。

5.3 列表条目元素

列表条目元素<li>，用来定义列表中的条目。语法结构如下所示。

```
<li>其他内容</li>
```

元素既可以使用在有序列表里，也可以使用在无序列表里。在以上的实例中，列表元素中均使用了<li>元素，所以就不做实例演示了。<li>元素中可以使用的所有属性如表 5-3 所示。

表 5-3 元素的所有属性

属 性 名 称	写 法
文本显示方向属性	dir
指定语言属性	lang
类属性	class
定义级联样式属性	style
项目符号的类型属性	type
标题属性	title
标记属性	id
条目编号属性	value

5.3.1 项目符号的类型属性 type

项目符号的类型属性 type 用来指定列表条目前的项目符号的类型。语法结构如下所示。

```
<li type="属性值">其他元素</li>
```

其中 type 属性的取值，根据包含<li>元素的父元素的类型不同，可以使用相应有序列表或无序列表中的值。

下面是一个在<li>元素使用 type 属性的实例。其代码如下所示。

例程 5-9 li-type.html

```
<body>
<font size=4 color="#990000">古典文学</font><br><br>
<ul>
  <li>唐诗</li>
  <li type="circle">宋词</li>
  <li>元散曲</li>
  <li type="square">明清小说</li>
  <li>经史子集</li>
</ul>
<hr size=2 color="#999999">
```

```
06  <font size=4 color="#990000">文化知识</font><br><br>
07  <ol>
     <li>经济知识</li>
     <li>文学艺术</li>
     <li type="A">科学文化</li>
     <li type="i">历史地理</li>
     <li type="a">政治哲学</li>
     <li>传统文学</li>
08  </ol>
09  </body>
```

在 03 行和 07 行的两个列表条目中，虽然用了不同的项目符号，但是代码运行后，每个项目符号都按顺序显示，所以第 07 行中第 3 个条目显示的是大写英文字母 C 而不是 A，第 4 个条目接着是罗马字母 iv，第 5 个条目接着是小写的英文字母 e，第 6 个条目默认，结果以最常用的阿拉伯字母 6 显示，代码运行后显示效果如图 5-9 所示。

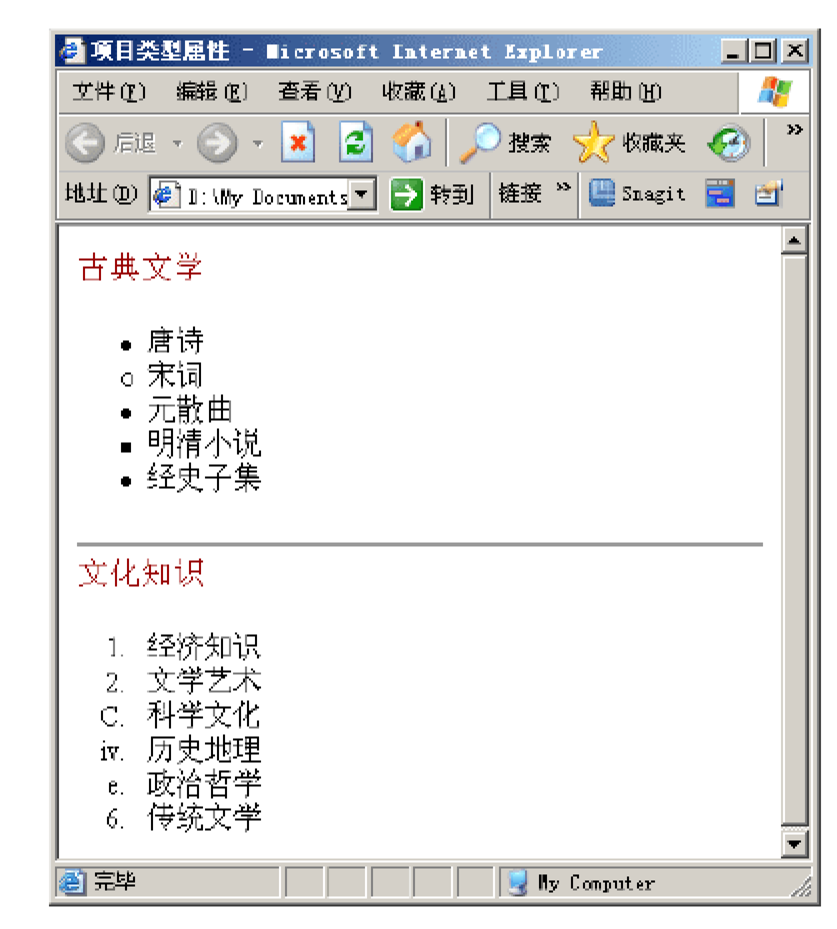


图 5-9　元素中使用 type 属性的显示效果

5.3.2　条目编号属性 value

条目编号属性 valu 用来更改有序列表中某一条目项目符号的初始值。同时，其后条目的项目符号也将随之改变。语法结构如下所示。

```
<ol>
      <li value="数字值">……</li>
</ol>
```

因为只有有序列表中才有条目的顺序问题，所以该属性只在<ol>元素中才能起作用。下面是一个使用 value 属性的实例。其代码如下所示。

例程 5-10　li-value.html

```
01  <body>
02  <ol>
     <li>列表内容 1</li>
     <li>列表内容 2</li>
     <li value="5">列表内容 3</li>
     <li>列表内容 4</li>
     <li>列表内容 5</li>
     <li>列表内容 6</li>
03  </ol>
04  </body>
```

在该实例中，使用 value 属性，将列表中的第 3 条编号更改为 5。其代码运行后，显示效果，如图 5-10 所示。

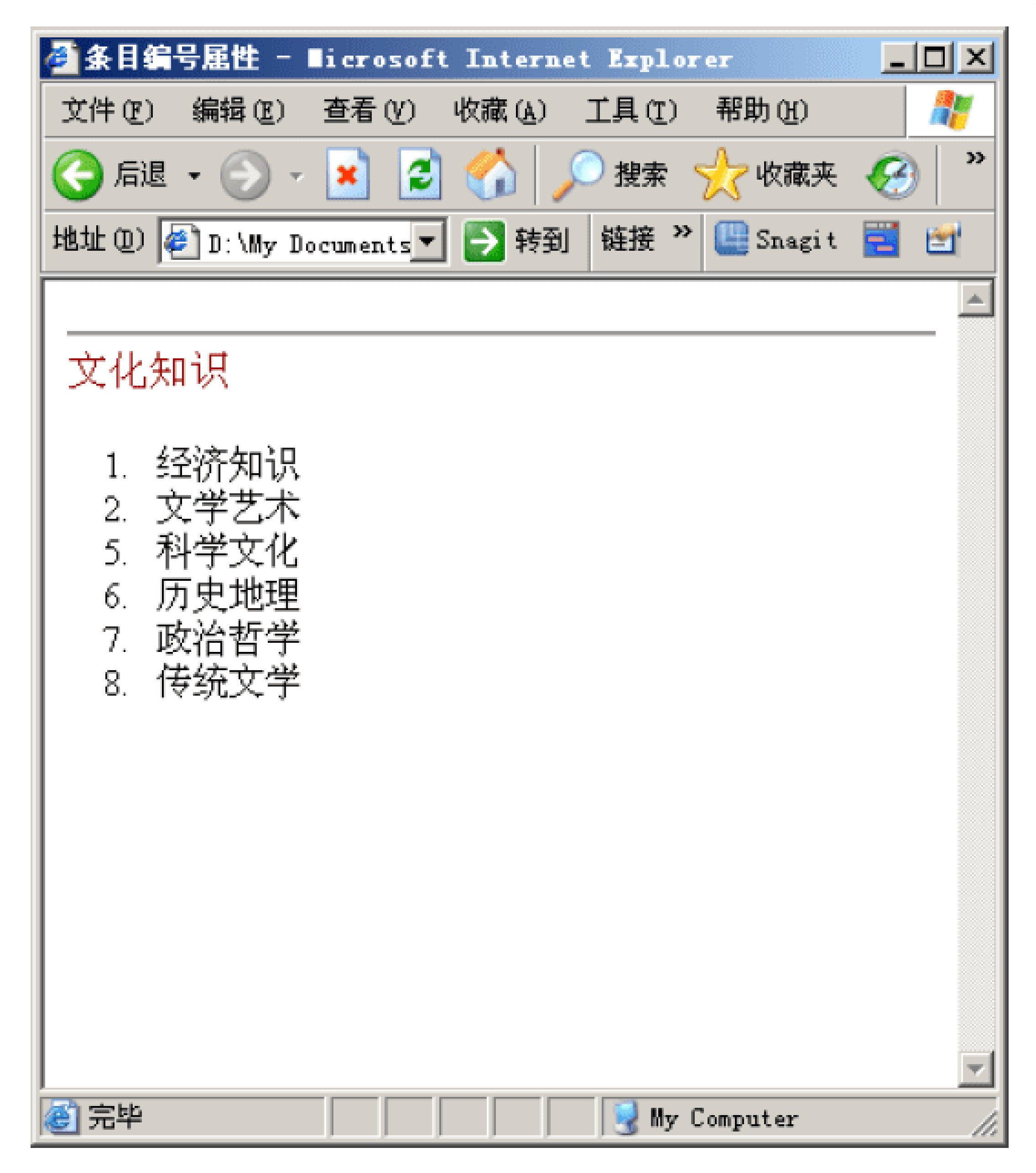


图 5-10 元素中使用 value 属性的显示效果

5.4 定义列表元素<dl>

定义列表元素<dl>，用来定义类似术语或者词汇表的列表信息。其中的具体条目信息，一般包括术语(或词汇等内容)，术语解释两个部分。语法结构如下所示。

```
<dl>其他的定义列表元素</dl>
```

下面是一个使用<dl>元素的实例。其代码如下所示。

例程 5-11 dl.html

```
01  <body>
02  <font size=6 color="#999999">简单的应用</font><br><br>
03  <dl>
     <dt>唐诗<dd>唐诗是我国古代特有的一种文学作品，一般有固定的字数，常见的有绝句和律诗两类。
     <dt>宋词<dd>宋词是继唐诗后我国又一类特有的文学作品。
     <dt>孙子兵法<dd>孙子兵法是我国古代著名兵法家孙武的军事著作，里面有很多对战略战术的精辟见解
     <dt>曹植<dd>“建安七子”之一，是我国三国时期的著名诗人。
04  </dl>
05  </body>
```

该实例中，03 行与 04 行之间简单地应用了<dl>元素的语法，其运行后的显示效果，如图 5-11 所示。

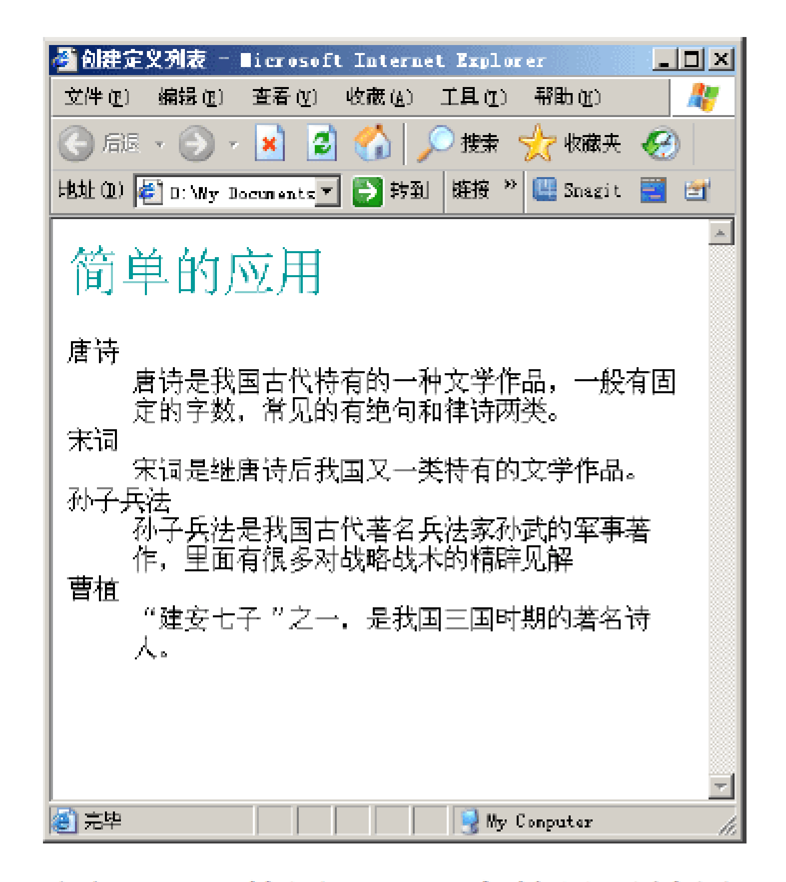


图 5-11　使用<dl>元素的显示效果

5.4.1　定义列表术语元素<dt>

定义列表术语元素<dt>用来定义<dl>元素中一个具体的条目。可以在<dt>元素中使用文本、图片等元素，但是不能使用其他的列表元素。语法结构如下所示。

```
<dl>
    <dt>……</dt>
</dl>
```

注意

<dt>元素可以在<dl>元素中单独使用，也可以在后面使用<dd>元素，一起构成定义列表。

下面是一个使用<dt>元素的实例。其代码如下所示。

例程 5-12　dt.html

```
<body>
<font size=6 color="#999999">唐代诗人</font><br><br>
<dl>
  <dt>陈子昂</dt>
  <dd>初唐最著名的诗人，开创了唐诗创作的新路。</dd>
  <dt>王维</dt>
  <dd>唐代著名田园诗人</dd>
  <dt>李白</dt>
  <dd>唐代著名浪漫主义诗人，有“诗仙”之称。</dd>
  <dt>杜甫</dt>
  <dd>唐代著名诗人，有“诗圣”之称</dd>
  <dt>王勃</dt>
  <dd>初唐四杰之首，《滕王阁序》是他的代表作。</dd>
</dl>
</body>
```

该实例中，在<dt>元素中使用文本。其运行后的显示效果，如图 5-12 所示。

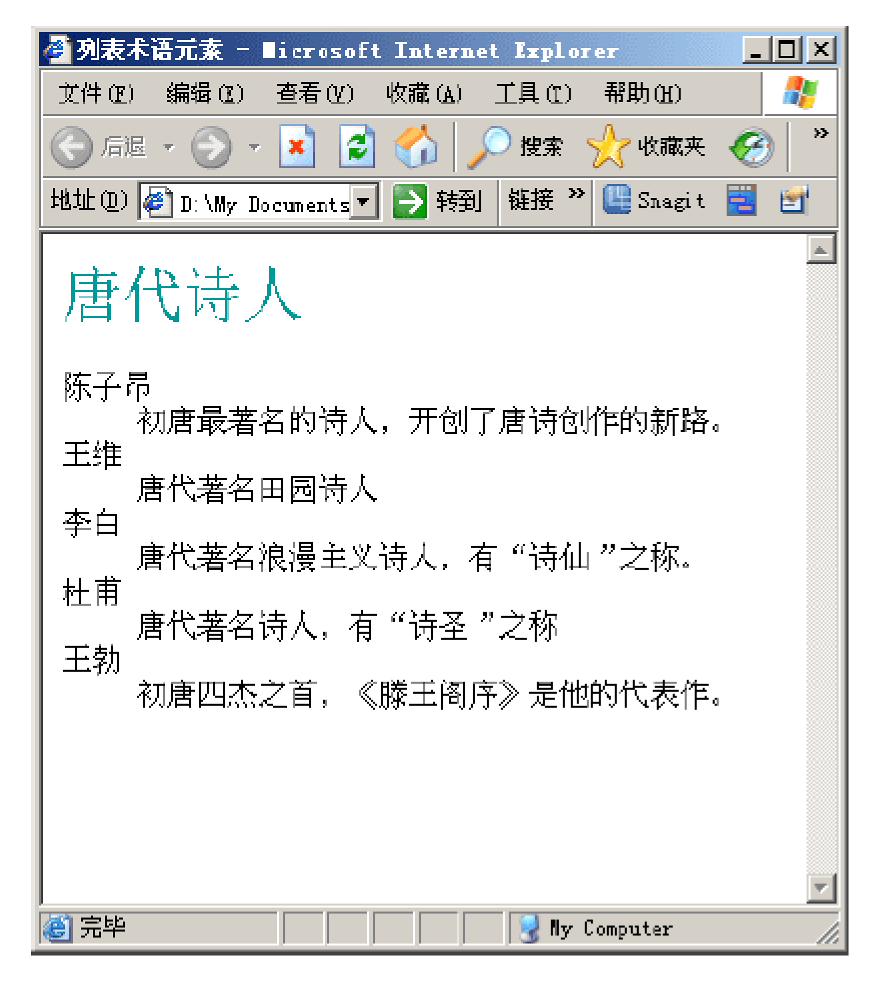


图 5-12　<dt>元素中使用文本元素的显示效果

5.4.2　定义列表条目说明元素<dd>

定义列表条目说明元素<dd>用来定义<dl>元素中一个具体的条目说明。语法结构如下所示。

```
<dl>
    <dd>其他元素</dd>
</dl>
```

注意

<dd>元素可以在<dl>元素中单独使用。同时可以在<dd>元素中使用其他的列表元素。

下面是一个使用<dd>元素的实例。其代码如下所示。

例程 5-13　dd.html

```
<body>
<font size=6 color="#999999">秦始皇</font><br><br>
<dl>
  <dt>主要功绩</dt>
  <dd>建立统一了的封建国家，结束了各国之间的无止休的征战。</dd>
  <dd>开创了中国封建制国家先河，是历史的一大进步。</dd>
  <dd>统一文字，促进了文化的发展。</dd>
  <dd>修建了万里长城</dd>
  <dd>实行郡县制，加强了国家的团结统一。</dd>
</dl>
</body>
```

其运行后的显示效果，如图 5-13 所示。

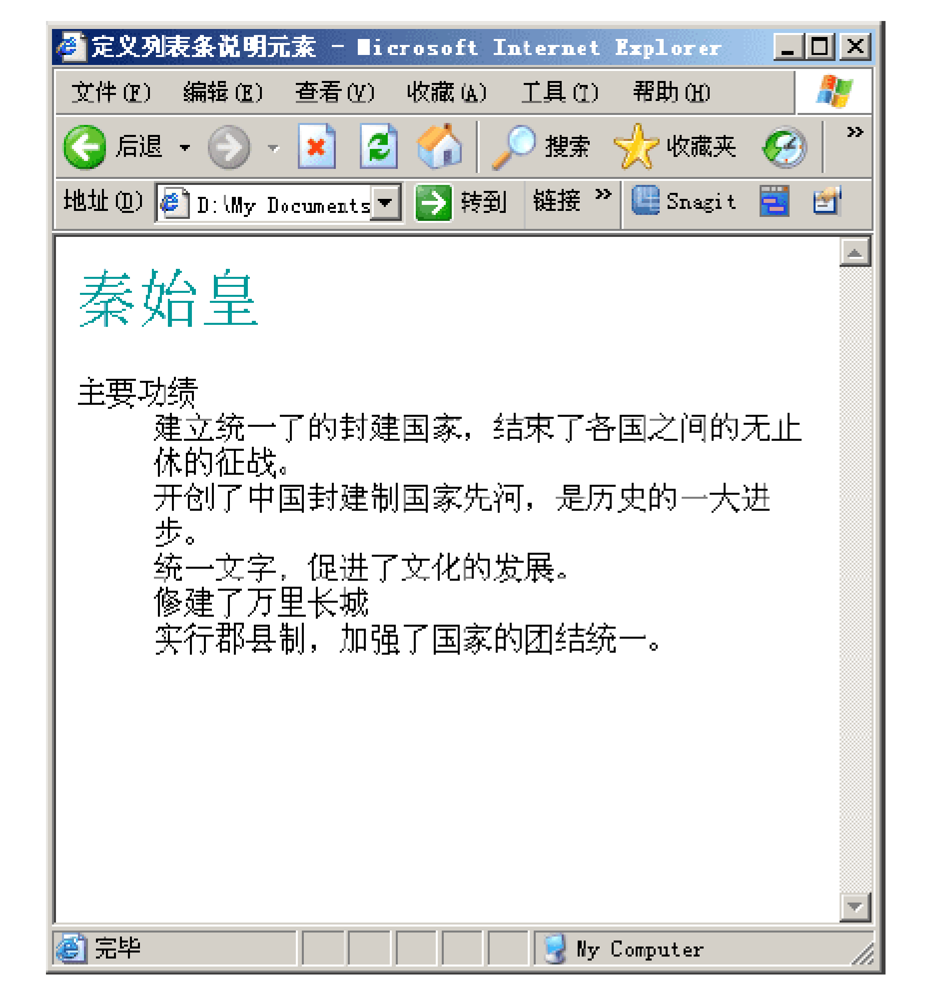


图 5-13　<dl>元素中使用<dd>元素的显示效果

5.5 本章习题

一、选择题

1．以下有关列表的说法中，错误的是(　　)。

A. 有序列表和无序列表可以互相嵌套

B. 指定嵌套列表时，也可以具体指定项目符号或编号样式

C. 无序列表应使用 UL 和 LI 标记符进行创建

D. 在创建列表时，LI 标记符的结束标记符不可省略

2．以下不是项目列表的前导符号的是(　　)。

A. 正方形　B. 实心圆　C. 空心圆　D. 菱形

二、填空题

1．列表元素包括__________和__________。

2. 在有序列表中，要求从 6 个大写罗马字母开始的代码是________________________。

三、实战练习

1．制作一个 6 条目的列表，要求每个列表前面用不同的符号。

2．制作一个 6 条目的列表，要求第 3 个条目从第 6 个大写英文字母 6 开始。

Chapter 6

图像元素

在浏览网页的时候，可以在网页上看到大量精美的图片，那么这些图片是如何放入页面的？在放入页面中时，应该如何处理？常见的图片有哪些类型，各有什么特点，将在本章中一一解答。希望通过本章的学习，读者能够掌握图像元素各种属性的使用方法和表现效果。

通过本章的学习，应该掌握以下知识要点：

本章主要内容有:

◎ 掌握把图片引入页面的方法。

◎ 能够简单地处理引入页面的图片。

◎ 了解图片的各种属性。

◎ 了解不同格式图片的特点。

6.1 图像元素<img>

图像元素<img>用来在页面中定义一个图片。由于<img>元素中引用的图片路径不同，可以使用各种格式的图片文件。语法结构如下所示。

```
<img 属性="属性值">……</img>
```

下面是一个使用<img>元素的实例。其代码如下所示。

例程 6-1　img.html

```
01  <body>
      <h3>西湖十景之雷峰塔</h3>
      雷峰塔简介<br>
      雷峰塔是杭州西湖的著名景点之一，我们熟悉的《白蛇传》中的故事就发生在这里，据说白娘子就是被压在这下面的，这里也是著名的西湖十景之一雷峰夕照。<br>
      <!--在页面中居中插入一张图片-->
02  <center>
03  <img src=" pic.jpg">
04  </center>
05  </body>
```

说明

在该语法中，src 参数用来设置图像文件所在的路径，这一路径可以是相对路径也可以是绝对路径。

该实例中，03 行中使用<img>元素调用了一个名称为 pic.jpg 的图片。其运行后的显示效果如图 6-1 所示。

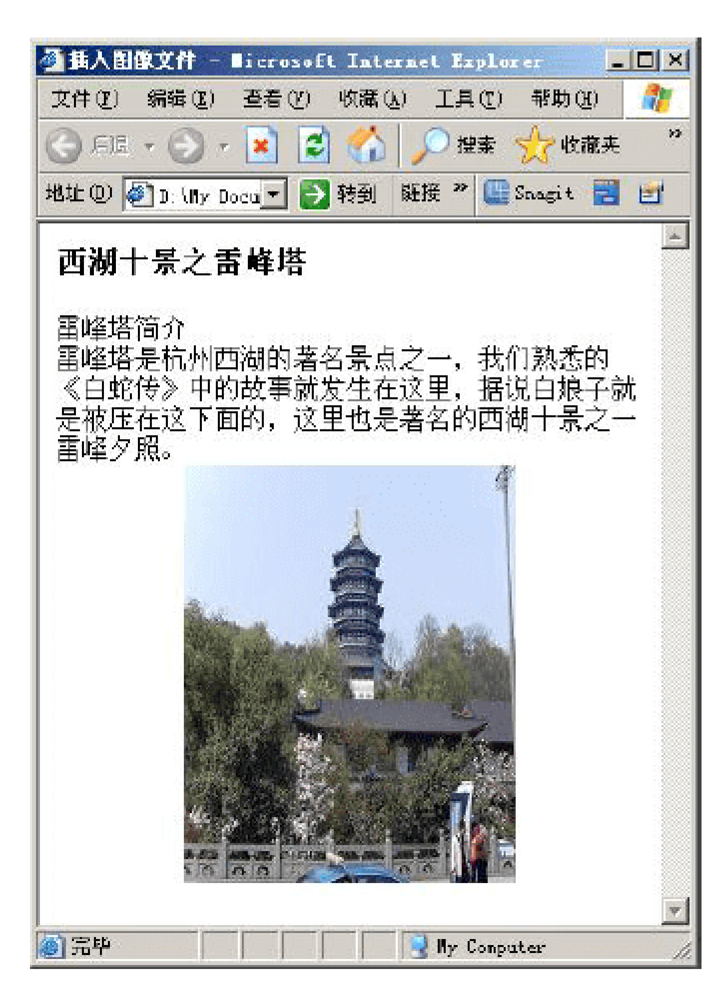


图 6-1　使用<img>元素的显示效果

<img>元素中可以使用的所有属性如表 6-1 所示。

表 6-1 <img>元素的所有属性

属 性 名 称	写 法
文本显示方向属性	dir
指定语言属性	lang
类属性	class
定义级联样式属性	style
对齐属性	align
标题属性	title
取消自动换行属性	nowrap
标记属性	id
路径属性	src
替代文本属性	alt
加长替代文本属性	langdesc
边框属性	border
高度属性	height
宽度属性	width
左右边距属性	hspace
上下边距属性	vspace
服务器端映射属性	ismap
客户端映射属性	usemap

6.1.1 图像元素的路径属性 src

图像元素的路径属性 src 用来定义一个调用图片的路径。src 属性是<img>元素中必须含有的属性，其中使用的图片格式没有限制，但是某些格式的图片可能无法正常显示。语法结构如下所示。

```
<img src="图片路径" />
```

其中，图片路径既可以是相对路径，也可以是绝对路径。下面是使用 src 属性的实例。代码如下。

例程 6-2 img-src.html

```
01 <body>
02 <img src="http://www.baidu.com/img/baidu_logo.gif" />
03 </body>
```

该实例中 02 行中调用了百度首页的 logo 图片。其运行后的显示效果如图 6-2 所示。



图 6-2　使用 src 属性的显示效果

6.1.2　代替图片的文本属性 alt

代替图片的文本属性 alt 用来定义当图片路径错误(或者其他原因无法显示时)，替代图片的文本。由于当鼠标悬停在图片内容上时，会显示 alt 属性中定义的文本，所以 alt 属性也可以用来定义图片的注释内容。语法结构如下所示。

```
<img src="图片路径" alt="文本内容" />
```

在 alt 属性中，可以使用的文本长度是 1024 个字符。下面是一个使用 alt 属性的实例。其代码如下所示。

例程 6-3　img-alt.html

```
01  <body>
02  <img src="http://www.baidu.com/img/baidulogo.gif"alt="请注意页面" />
03  </body>
```

在该实例中，02 行中将调用百度首页的 logo 图片的路径写错了。其运行后的显示效果如图 6-3 所示。当用户运行这个例程代码后，把鼠标放在图片的位置就能看到写有“请注意页面”的字样了。

下面是一个使用 alt 属性制作图片注释的实例。其代码如下所示。

例程 6-4　img-alttwo.html

```
01  <body>
02  <img src=" http://www.cctv.com/homepage2006/images/cctvlogo_778.gif " alt="这里是CCTV 的logo 图片" />
03  </body>
```

在该实例中，02 行中使用 alt 属性为图片定义注释内容。运行后，当鼠标悬停在图片上时，

显示效果如图 6-4 所示。





图 6-3　使用 alt 属性的显示效果



图 6-4　使用 alt 属性定义的注释效果

6.1.3　图像元素的宽度属性 width

在网页中直接插入图片的时候，图像的大小和原图是相同的，而在实际应用的时候需要通过各种图像属性的设置调整图像的大小、分辨率等内容。

用图像元素的宽度属性 width 来定义图片元素的宽度。语法结构如下所示。

```
<img src="图片路径" alt="文本内容" width="数字值" />
```

其中数字值的实际单位是像素。下面是一个使用 width 属性的实例。其代码如下所示。

例程 6-5　img-width.html

```
01	<body>
02	<h3>雷峰塔</h3>
03	<hr size=3>
	<br>雷峰塔原建造在雷峰上，位于杭州西湖南岸南屏山日慧峰下净慈寺前。雷峰为南屏山向北伸展的余脉，濒湖 勃然隆起，林木葱郁。但民间因塔在雷峰之上，均呼之为雷峰塔。原塔共七层，重檐飞栋，窗户洞达，十分壮观。
	<br>
<!--在页面中居中插入两张图片-->
04	<center>
	<!--默认的图片大小-->
05	<img src=" pic.JPG">
	<!--设置图片的高度为 160 像素-->
06	<img src=" pic.JPG" width=160>
07	</center>
08	</body>
```

在该实例中，在页面中插入图片时，05 行是按默认自动插入，页面中图片大小和原图一

样，06 行是定义图片宽度为 160 像素。其运行后的显示效果如图 6-5 所示。从图 6-5 可以看出，虽然只定义了图片的宽度，但是图片的高度也会按照原图片的比例相应调整。



图 6-5　使用 width 属性定义图片宽度后的效果

6.1.4　图像元素的高度属性 height

图像高度的属性与图像宽度属性类似，同样是用来调整图像的大小的。可以用 height 来定义图片元素的高度。语法结构如下所示。

```
<img src="图片路径" alt="文本内容" height="数字值" />
```

在该语法中，图像的宽度单位是像素。如果在使用属性的过程中，只设置了高度或宽度，则另外一个参数会等比例变化。如果同时设置两个属性，且缩放比例不同的情况下，图像很可能会变形，下面实例同时使用 width 属性和 height 属性。其代码如下所示。

例程 6-6　img-height.html

```
01  <body>
     <h3>雷峰塔</h3>
     <hr size=3>
         <br>雷峰塔原建造在雷峰上，位于杭州西湖南岸南屏山日慧峰下净慈寺前。雷峰为南屏山向北伸展的余脉，濒湖勃然隆起，林木葱郁。但民间因塔在雷峰之上，均呼之为雷峰塔。原塔共七层，重檐飞栋，窗户洞达，十分壮观。<br>
02  <center>
03  <img src=" pic.JPG" height=160>
04  <img src=" pic.JPG"width=160 height=160>
05  </center>
06  </body>
```

在该实例中，03 行中定义第一张图片的高度为 160 像素，04 行中第二张图片的高度和宽度都为 160 像素。其运行后的显示效果，如图 6-6 所示。第一张图只设置高度，宽度默认，所以在页面中按原图比例自动调整。第二张图片由于在<img>元素中，定义的高度和宽度与原图片的比例不相同，所以图片产生了变形的现象，在图 6-6 中以正方形图片出现在页面中。

定义了图片的宽度和高度后，如果图片不能显示的时候，代替图片的文本，会出现在图片宽度和高度定义的区域内。并且当 alt 属性中定义的文本内容高度大于图片大小时，超出的部分将被隐藏。其实例代码如下所示。

例程 6-7　img-heighttwo.html

```
<body>
```

```
        <img src="pic.jpg" alt="《麦田里的守望者》故事讲述的主人公是 16 岁的中学生霍尔顿·考尔德是当代美国文学中最早出现的反英雄形象之一，霍尔顿出身中纽约一个富裕的中产阶级家庭。学校里的老师和自己的家长强迫他好好读书，为的是出人头地，以便将来买辆凯迪拉克，而在学校里一天到晚干的，就是谈女人，酒和性，他看不惯周围的一切，根本没心思用功读书，因而老是挨罚，到他第四次被开除时，他不敢回家。便只身在美国最繁华的纽约城游荡了一天两夜，住小客店，逛夜总会滥交女友他在电影院里百无聊赖地消磨时光，糊里糊涂地召了妓女，情不自禁的与虚荣的女友搂搂抱抱，与此同时，他的内心又十分苦闷，企图逃出虚伪的成人世界去寻批纯洁与真理的经历与感受。这种精神上无法调和的极度矛盾最终令他彻底崩溃，躺倒在精神病院里......
        作者塞林格全名杰罗姆·大卫·塞林格，一九一九年生于美国纽约城，父亲是做于酪和火腿进口生意的犹太商人，家境相当富裕。塞林格十五岁的时候，被父母送到宾夕法尼亚州一个军事学校里住读，据说《麦田里的守望者》中关于寄宿学校的描写，很大部分是以那所学校为背景的。一九三六年，塞林格在军事学校毕业，取得了他毕生唯一的一张文凭。        从一九四〇年在《小说》杂志上发表他的头一个短篇小说起，到一九五一年出版他的长篇小说《麦田里的守望者》止，在十余年中他共发表了二十多个短篇，有些短篇还在《老爷》、《纽约人》等著名刊物上发表，从而使他在文学界有了一点点名气。成名后他隐居到乡下，特地为自己造了一个只有一扇天窗的水泥斗室作书房，每天早晨八点半就带了饭盒入内写作，直到下午五点半才出来，家里任何人都不准进去打扰他；如有要事，只能用电话联系。他写作的过程据说还十分艰苦，从《麦田里的守望者》出版后，他写作的进度越来越慢，十年只出版三个中篇和一个短篇，后来甚至不再发表作品。偶尔有幸见过他的人透露说，  " width="300" height="300" />
        </body>
```

该代码运行后，显示效果如图 6-7 所示。



图 6-6　图片同时定义了高度和宽度后的效果

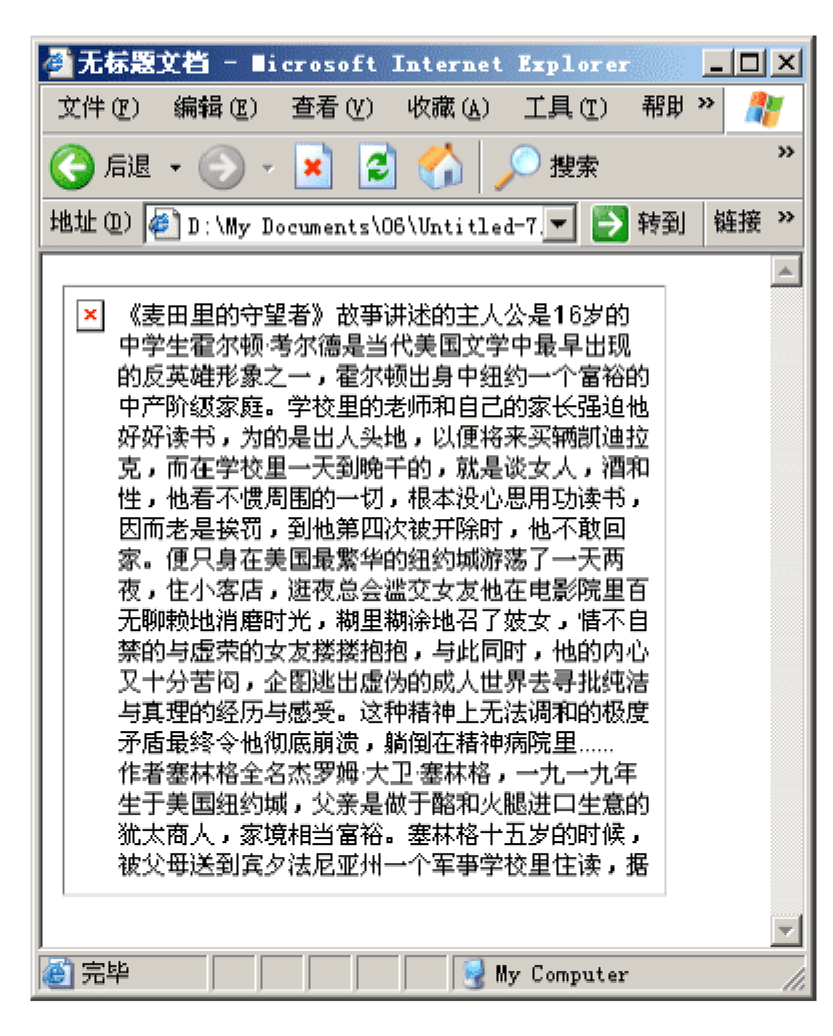


图 6-7　当含有大量替换文本时的显示效果

6.1.5　图像元素的边框属性 border

在制作图像的时候，一般用图像元素的边框属性 border 来定义图片的边框。普通的图片元素在显示时是没有边框的，可以通过 border 属性为图像添加边框。语法结构如下。

```
<img src="图像文件的地址" border="图像边框的宽度">
```

在该语法中，src 是图像文件的地址，是不可缺少的。border 的单位是像素。下面一起来看看使用 border 元素的实例，其代码如下所示。

例程 6-8 img-border.html

```
01	<body>
	 <h3>狼(Beauty and the Beast)</h3>
	 <hr size=3>
	 狼是自然界进化的最成功的动物，也是最有纪律和团队合作精神的动物。狼有很强的合作意识和团队合作精神，这些都值得我们学习。
	<br>
02	<center>
03	<img src="狼.jpg">
04	<img src="jpg" border="15">
05	</center>
06	</body>
```

该实例中，03 行中先直接引用了一张图片做对照，04 行为了使图片的边框显示得更加明显，定义了一个很粗的边框。代码运行后，其显示效果如图 6-8 所示。

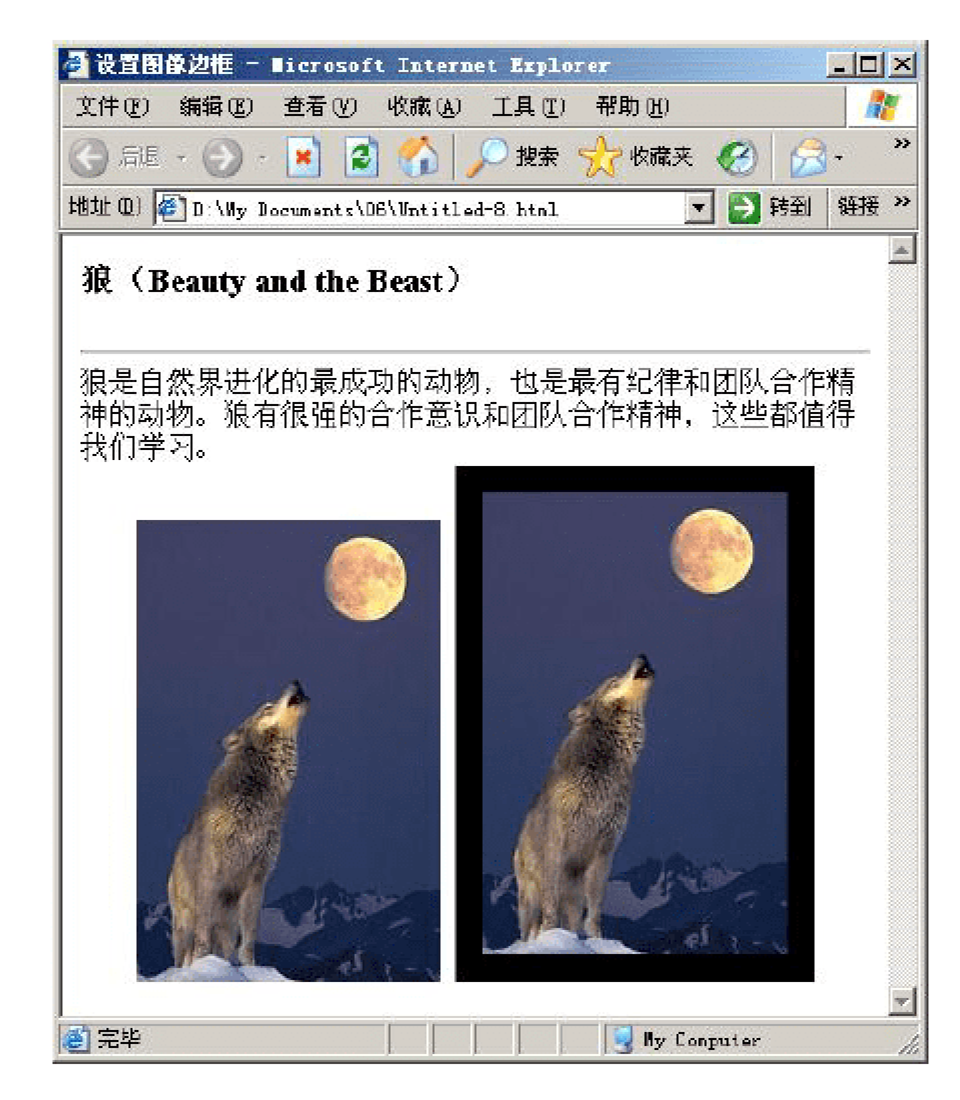


图 6-8　图片定义 border 属性后的显示效果

6.1.6　代替图片的长文本属性 longdesc

有时候要用文本来代替图片。代替图片的长文本属性 longdesc 用来定义代替图片的文本。langdesc 属性和 alt 属性的功能基本相同，也可以用来定义图片的注释。语法结构如下所示。

```
<img src="图片路径" langdesc="替代文本路径" />
```

当代替文本长度超过 1024 字符时，可以使用 longdesc 属性，定义一个链接文件来显示替代文本。目前常用的浏览器还不支持该属性。因为浏览器还不支持该属性，所以就不做实例讲解了。

6.1.7　上下边距属性 vspace

用图像元素的上下边距属性 vspace 来使图片元素与相邻的内容在垂直方向上分开一段距离。语法结构如下所示。

```
<img src="图片路径" alt="文本内容" vspace="上下距离" />
```

其中数字值的实际单位是像素。下面是一个使用 vspace 属性的实例。其代码如下所示。

例程 6-9 img-vspace.html

```
<body>
<img src="狼.jpg" vspace="50">
 狼是群居性极高的物种。一群狼的数量大约在 5 到 12 只之间，在冬天寒冷的时候最多可到四十只左右，通常以家庭为单位的家庭狼由一对优势对偶领导，而以兄弟姐妹为一群的则以最强一头狼领导。狼群有领
```

```
域性，且通常也都是其活动范围，群内个体数量若增加，领域范围会缩小。群之间的领域范围不重叠，会以嚎声向其他群宣告范围。幼狼成长后，会留在群内照顾弟妹，也可能继承群内优势地位，有的则会迁移出去(大都为雄狼)而还有一些情况下会出现迁徙狼，以百来头为一群，有来自不同家庭等级的各类狼，各个小团体原狼首领会成为头狼，头狼中最出众的则会成为狼王。野生的狼一般可以活 12——16 年，人工饲养的狼有的可以活到二十年左右。 奔跑速度极快，可达五十五公里左右，持久性也很好。它们有能力以速度 10 公里/小时(六英里)长时间奔跑 ，并能以高达近 65 公里/小时速度( 40 英里) 追猎冲刺。 如果是长跑，它的速度会超过猎豹。智能颇高，可以气味、叫声沟通。 狼是以肉食为主的杂食性动物,是生物链中极关键的一节。
03   </body>
```

在该实例中，02 行中定义了上边图片元素的上下边距属性值为 50 像素。代码运行后，图片和文字之间就会有 50 像素的差距。其显示效果如图 6-9 所示。

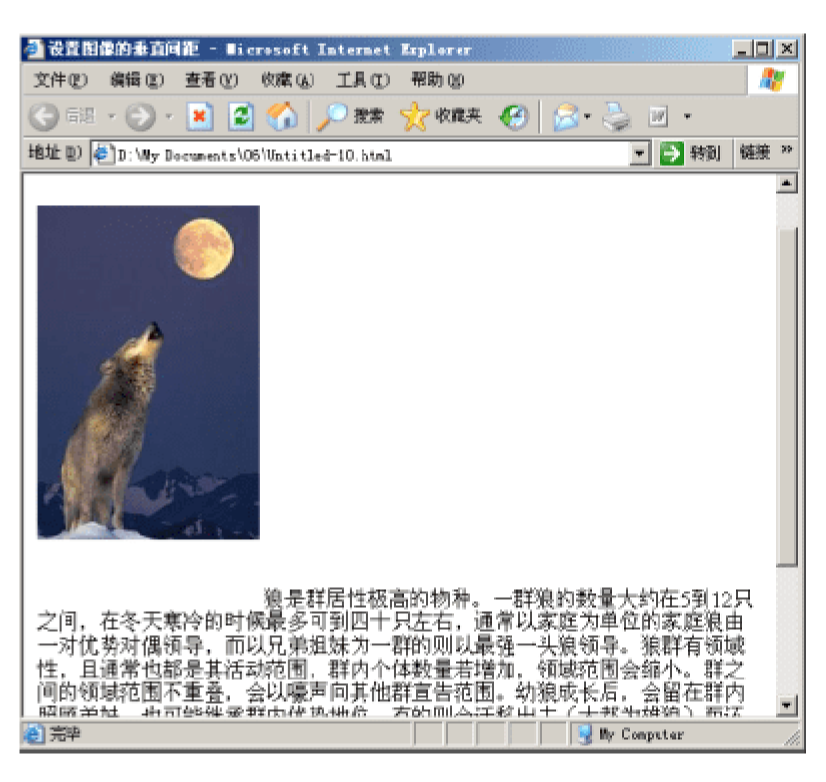


图 6-9　设置上下边距后的效果

6.1.8　图像元素的左右边距属性 hspace

图像元素的左右边距属性 hspace 用来使图片元素与相邻的内容，在水平方向上分开一段距离。语法结构如下所示。

```
<img src="图片路径"   hspace="数字值" />
```

其中数字值的实际单位是像素。下面是一个使用 hspace 属性的实例。其代码如下所示。

例程 6-10　img-hspace.html

```
01   <body>
       <h3>狼</h3>
       <hr size=2>
       中国是狼种群数量最大的国家之一。
02   <img src="狼.jpg" border="4">
       <br><br>
       中国曾是狼种群数量最大的国家之一.
03   <img src="狼.jpg" border="4" hspace="50">
04   </body>
```

在该实例中，02 行中第一张图和文字之间的距离为默认，03 行中第二张图中定义了图片和文字的左右边距属性值为 30 像素。代码运行后，其显示效果如图 6-10 所示。

从图 6-10 中可以看出下图的文字和图片之间有较为明显的距离。

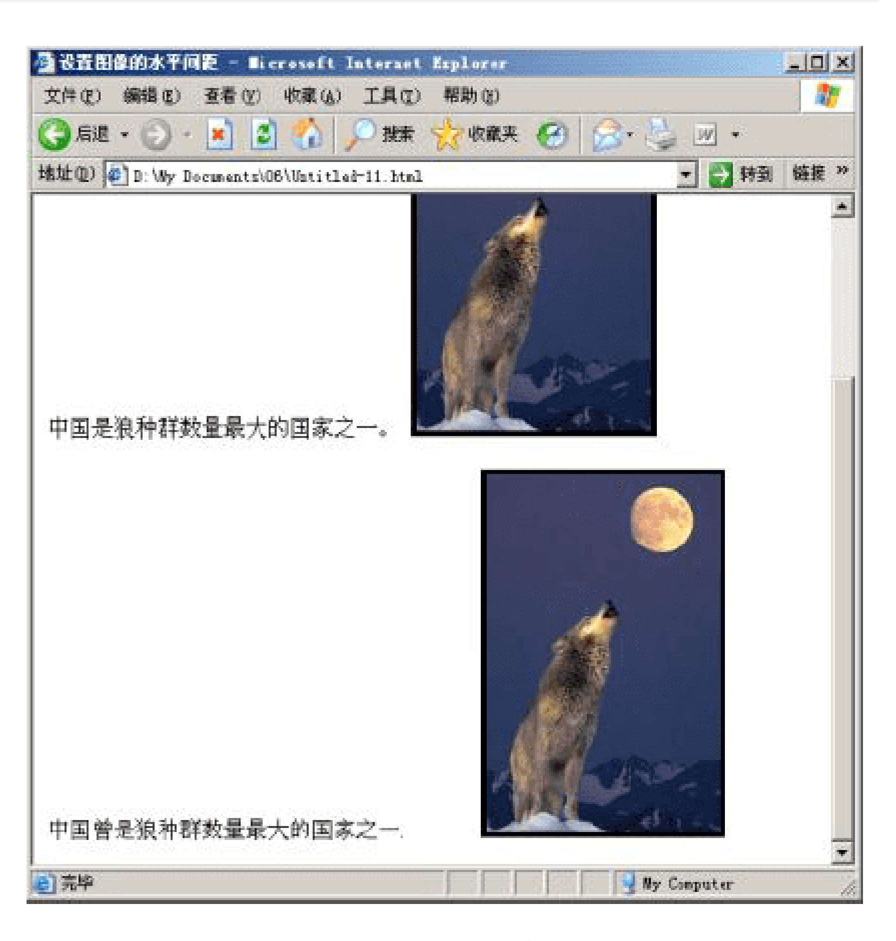


图 6-10　定义 hspace 属性后的显示效果

6.1.9　图像元素的对齐属性 align

图像和文字之间的排列方式可以通过图像元素的对齐属性 align 来调整。图像的绝对对齐方式与相对文

字的对齐方式不同，绝对对齐方式包括左对齐、右对齐和居中对齐 3 种，而相对文字对齐方式则是指图像与一行文字的相对位置。语法结构如下所示。

```
<img src="图像文件的地址" align="相对文字的对齐方式">
```

注意

align 属性的取值比较复杂。其可以分为 3 个部分，一部分是标准中规定的值；另一部分是 Netscape 增加的值；还有一部分是 IE 中增加的值。标准中规定的值如表 6-2 所示。

表 6-2　align 属性的取值及含义

align 取值	表示的含义
top	把图像的顶部和同行的最高部分对齐(可能是文本的顶部，也可能是图像的顶部)
middle	把图像的中部和行的中部对齐(通常是文本行的基线，并不是实际的行的中部)
bottom	把图像的底部和同行文本的底部对齐
texttop	把图像的顶部和同行中最高的文本的顶部对齐，常用于 Netscape 中
absmiddle	把图像的中部和同行中最大项的中部对齐，常用于 Netscape 中
baseline	把图像的底部和文本的基线对齐，常用于 Netscape 中
absbottom	把图像的底部和同行中的最低项对齐，常用于 Netscape 中
left	使图像和左边界对齐(文本环绕图像)
right	使图像和右边界对齐(文本环绕图像)

下面分别通过实例，演示各个属性的显示效果。其代码如下所示。

例程 6-11　img-align.html

```
01  <body>
     <font size=4>作为基准的文字</font>
     <!--图像的底端与文字的底端对齐-->
     <img src="狼.jpg" width=75 align="bottom">
     <!--图像的中间与文字的中间线对齐-->
     <img src="狼.jpg" width=75 align="texttop">
     <!--图像的顶端与文字的顶端对齐-->
     <img src="狼.jpg" width=75 align="middle">
     <!--图像的中间线与文字的中间线对齐-->
     <img src="狼.jpg" width=75 align="baseline">
     <!--图像的底端与文字的底端对齐-->
     <img src="狼.jpg" width=75 align="absmiddle">
    <br>
02  <hr size=2>
     下面是图像位于文字左侧的效果：<br><br>
03  <img src="狼.jpg" width=75 align="left">这里讲解的是关于图像与文字的相对位置的设置，文字作为图
```

```
像对齐效果的参照物。如果将图像的对齐方式设置成 left 或者 right，图像则会与文字进行环绕显示。这是图像设置为 left 的效果。<br><br><br>
        下面显示图像位于文字右侧的效果：<br><br>
04  <img src="狼.jpg" width=75 align="right">这里讲解的是关于图像与文字的相对位置的设置，文字作为图像对齐效果的参照物。如果将图像的对齐方式设置成 left 或者 right，图像则会与文字进行环绕显示。这是图像设置为 right 的效果。
    <br></br>
05  </body>
```

01 行中条目定义了文字与图像的相对垂直位置的变化效果，02 行为水平线，03 行中定义左对齐，04 行定义了右对齐的效果。代码运行后，显示效果如图 6-11 所示。

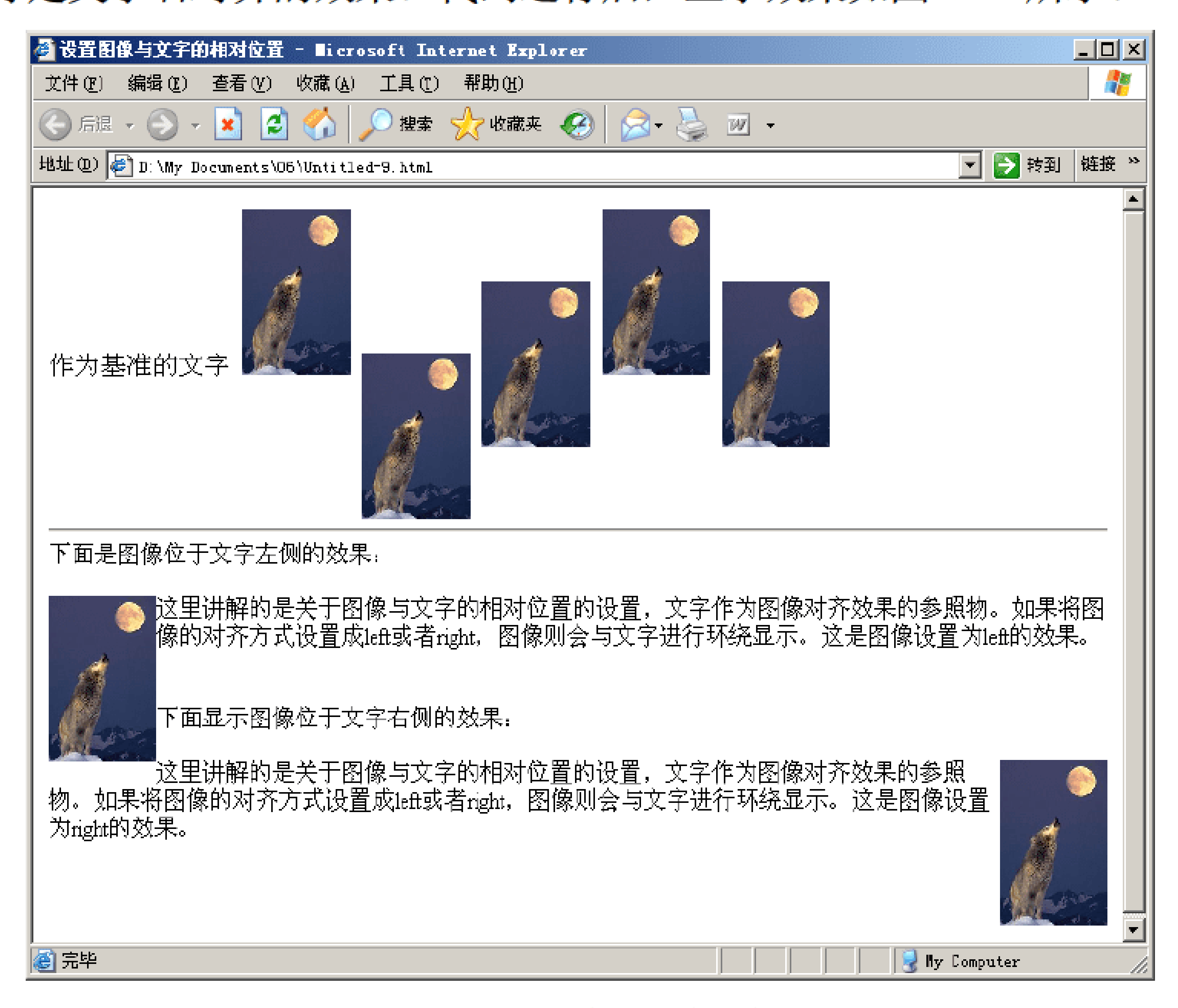


图 6-11　图像的对齐属性

6.1.10　图像服务器端映射属性 ismap

用图像服务器端映射属性 ismap 来将图像的坐标传递给相应的服务器端处理程序。但是注意 ismap 只可以用在 <a> 标签标识的超链接里面。语法结构如下所示。

```
<a href="相应的程序">
   <img src="图片路径" ismap=ismap />
</a>
```

注意<img>元素必须使用在<a>元素之中。

下面是一个 ismap 属性的实例。其代码如下所示。

例程 6-12　img-ismap.html

```
01  <body>
02  <a href="link.cgi">
```

```
03 <img src="狼.jpg"   ismap="ismap" />
04 </a>
05 </body>
```

03 行中使用了图像服务器端映射属性 ismap，当用户在图像上单击了某处时，浏览器会自动把鼠标的 x、y 位置(相对于图像的左上角)发送到服务器端。特殊的服务器端软件(在本例中是 link.cgi 程序)可以根据这些坐标来做出响应。

6.1.11 图像服务器端映射属性 usemap

图像服务器端映射属性 usemap 用来通过相应的<map>和<area>元素，在图片的相应区域定义一个超链接(关于<map>和<area>元素以及 usemap 属性的详细应用，请参照第 8 章)。语法结构如下所示。

```
<img src="图片路径" alt="图片" usemap="#map" />
```

usemap 属性必须和<map>元素等一起使用。下面是一个使用 usemap 属性的实例。其代码如下所示。

例程 6-13 img-usemap.html

```
<body>
<img src="pic.jpg" alt="图片" usemap="#Map" />
</body>
```

6.2 图像的格式

在网页中经常能看到大量的精美图片。随着现在数码技术的发展，人们也经常用数码相机拍照，当把数码相片移动到电脑中时，用鼠标右击属性，就会看到图片格式的说明。常见的图片格式有 JPEG 格式、GIF 格式和 PNG 格式。这些格式在制作网页的时候经常用到，下面一起了解它们的特点。

6.2.1 JPEG 格式

JPEG 格式的图片是最常见也是使用最多的图片，人们用数码相机照出来的图片一般就是这种格式。这种格式的图片的好处在于各种浏览器都能很好的支持，同时可以方便地压缩图片的大小，还可以很好地显示颜色复杂、质量精细度要求很高的图片内容。但是在处理大面积的颜色块时，这种图片可能会出现明显的压缩痕迹。

JPEG 格式的图片的图片用途很广，在网页中看到的大部分图片都是这种格式，下面是一个 JPEG 格式的图片，如图 6-12 所示。JPEG 格式的图片的后缀名是“.jpg”。

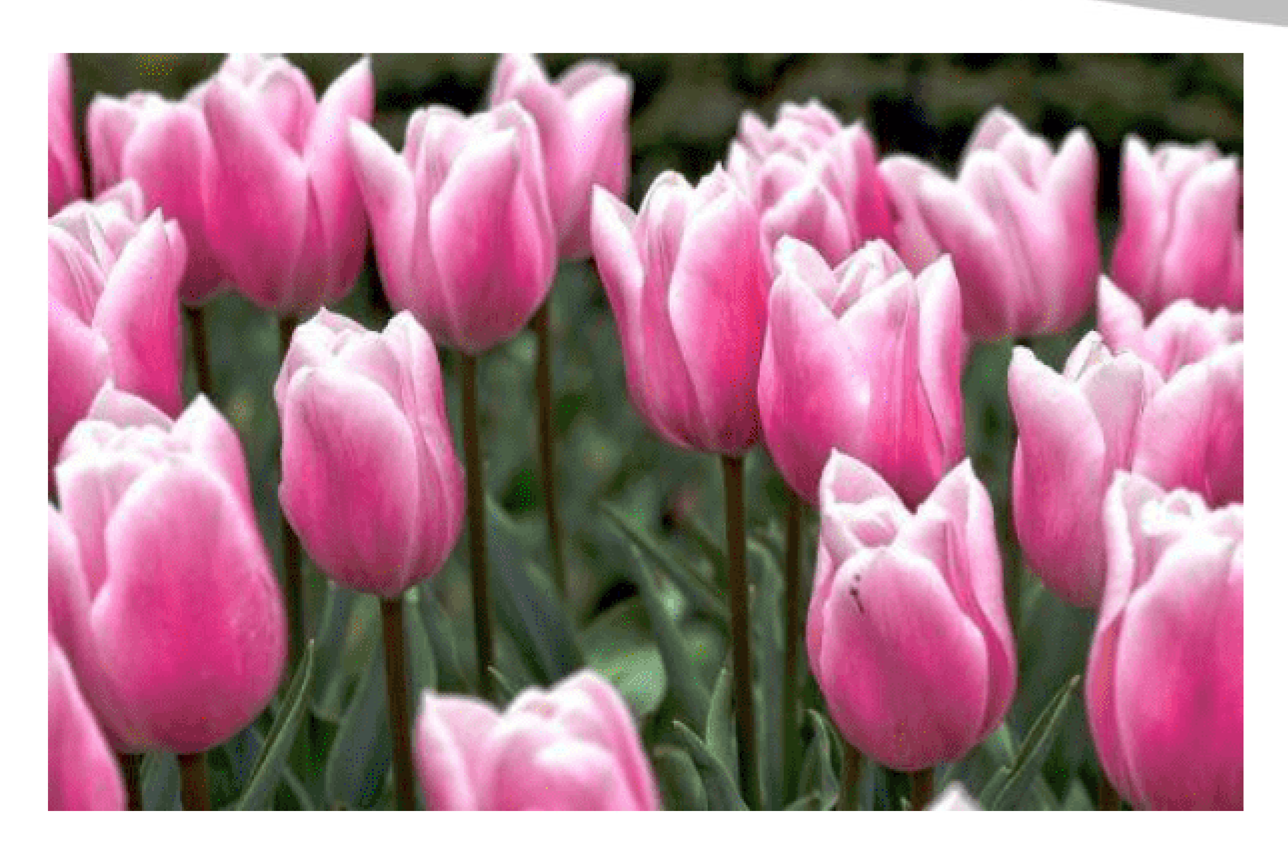


图 6-12　JPEG 格式的图片

6.2.2　GIF 格式

GIF 格式即图形交换格式(Graphics Interchange Format 的简写)。这种格式的图片在任何浏览器中均能正常显示，所以在网页设计中使用的最多。不过 GIF 格式的图片只能够使用 256 种色彩，所以不适合显示色彩比较丰富的图像内容(如照片等)。GIF 格式的图片具有 3 个比较突出的特性。

1．可以设置背景透明

GIF 格式的图片可以设置背景的透明。它不像其他格式的图片一样显示带有白色背景的矩形框。因而可以显示不规则的图形，这一点在制作网页时非常有用，可以通过透明的背景格式，设置页面中的图标、logo 等。

2．可以制作简单的动画

可以使用相应的工具制作简单的 GIF 格式的动画，这种格式的动画不需要安装任何多余的插件即可正常显示，而且可以方便地放置在页面的任何位置。

3．采用隔行扫描的显示方式

GIF 格式的图片由于具有隔行扫描的效果，所以在显示时，不会像 JPEG 等其他格式的图片一样，从上到下像打开一个卷轴一样显示出来，而是像打开百叶窗一样显现出来，同时会出现一种从模糊到清晰的显示效果。其在显示速度上有着明显的优势。

但是使用 GIF 格式也会带来一些问题，就是图片文件的大小会增加，过多地使用 GIF 格式的动画，可能会增加页面加载的时间。下面是一个 GIF 格式的动画图片，如图 6-13 所示。GIF 格式的图片的后缀名为“.gif”。

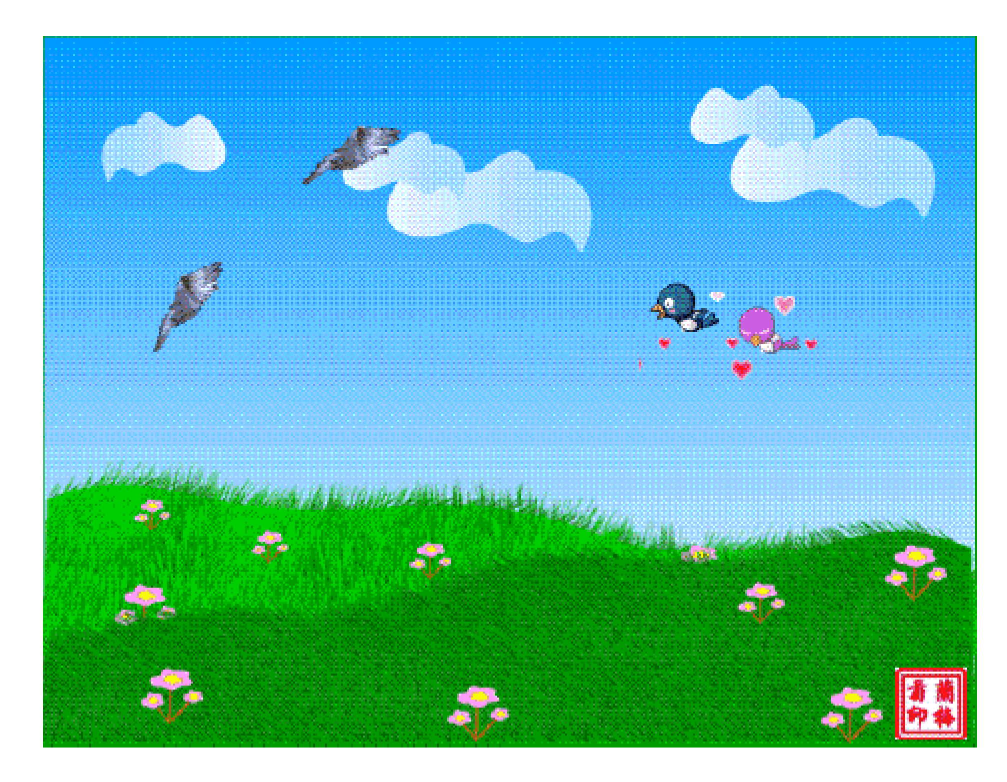


图 6-13　GIF 格式的图片

6.2.3 PNG 格式

PNG 格式的图片具有压缩比高，生成文件容量小的优点。因而能很好地保证图片不失真，并且具有 GIF 和 JPG 二者的优点，存储形式丰富，兼有 GIF 和 JPG 的色彩模式。

PNG 格式图片的另一个特点就是能把图像文件压缩到极限以利于网络传输，但又能保留所有与图像品质有关的信息，因为 PNG 是采用无损压缩方式来减少文件的大小，这一点与牺牲图像品质以换取高压缩率的 JPG 有所不同。

它的第 3 个特点是显示速度很快，只需下载 1/64 的图像信息就可以显示出低分辨率的预览图像；不过这种格式的图片不支持动画应用效果，因而它的应用受到一定的限制。下面是一个 PNG 格式的动画图片，如图 6-14 所示。PNG 格式的图片的后缀名为“. PNG”。



图 6-14　PNG 格式的图片

6.3 本章习题

一、选择题

1．下面对 JPEG 格式描述不正确的一项是(　)。

A. 照片、油画和一些细腻、讲求色彩浓淡的图片常采用 JPEG 格式

B. JPEG 支持很高的压缩率，因此其图像的下载速度非常快

C. 最高只能以 256 色显示的用户可能无法观看 JPEG 图像

D. 采用 JPEG 格式对图片进行压缩后，还能再打开图片，然后对它重新整饰、编辑、压缩

2．有关网页中的图像的说法不正确的是(　)。

A. 网页中的图像并不与网页保存在同一个文件中，每个图像单独保存

B. HTML 语言可以描述图像的位置、大小等属性

C. HTML 语言可以直接描述图像上的像素

D. 图像可以作为超级链接的起始对象

3．以下关于 JPEG 图像格式中，错误的是(　)。

A. 适合表现真彩色的照片。

B. 最多可以指定 1024 种颜色。

C. 不能设置透明度。

D. 可以控制压缩比例。

二、填空题

1．在网页中插入背景图案(文件的路径及名称为/img/bg.jpg)的语句是____________。

2．设置文字的颜色为红色的标记格式是_________。

3．设置颜色可以用颜色的英文名称，也可用_________。

4．插入图片<img src="图形文件名">标记符中的 src 英文单词是________。

5．设定图片边框的属性是__________。

6．设定图片高度及宽度的属性是______________。

7．设定图片上下留空的属性是__________；设定图片左右留空的属性是__________。

8．为图片添加简要说明文字的属性是_______。

9．已知站点文件夹结构如图 6-15 所示，要在 interest.htm 这个网页中插入 sunset.gif，应使用语句___。

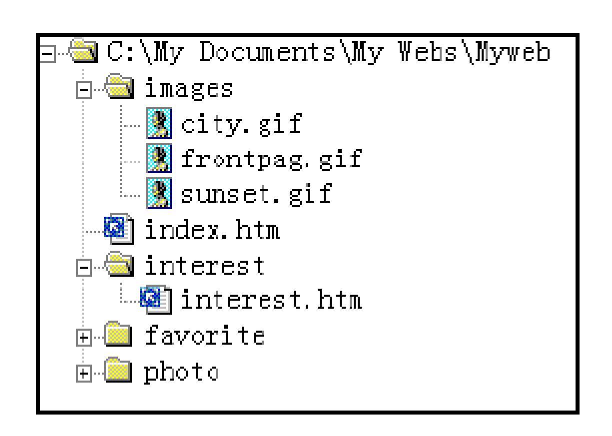


图 6-15　站点文件夹结构图

三、实战练习

1．在页面链接一张图片，并设置它的边框、高度和宽度。

2．选择一个图片，并用在页面中添加一条水平线来做基准，设置图片的各种对齐属性。

3．选择一个图片，并在页面中设置它的边距属性。

Chapter 7

表格元素

在制作页面的过程中，会用到各种表格来对页面进行排版。如何把表格美观、实用地放在要制作的页面中呢？本章将讲解表格元素及其相关元素的属性和应用。

本章主要内容有:

◎ 重点掌握表格元素各种属性的特点和用法。

◎ 能够按照页面的要求制作表格。

◎ 掌握表格的边界属性的各个特点。

◎ 通过本章学习能熟练制作表格。

7.1 表格元素的结构

与其他元素不同，在表格中一般通过 3 个标记来构建，分别是表格标记、行标记和单元格标记。其中表格标记是<table>和</table>，表格的其他各种属性都要在表格的开始标记<table>和表格的结束标记</table>之间才有效。下面首先介绍如何创建表格，其语法如下所示。

```
<table>
  <tr>
    <td>……</td>
  </tr>
</table>
```

该语法中<table>标记和</table>标记分别标志着一个表格的开始和结束；而<tr>和</tr>则分别表示表格中一行的开始和结束，在表格中包含几组<tr>…</tr>，就表示该表格为几行；<td>和</td>表示一个单元格的起始和结束，也可以说表示一行中包含了几列。

表格元素在没有定义任何属性时，是没有表现效果的。例如只用语法结构中定义的表格，在页面中显示其效果如图 7-1 所示。



图 7-1　表格元素没定义属性时的显示效果

下面对表格元素中必须使用的各个具体元素进行详细讲解。

7.1.1　表格<table>

表格元素<table>用来在页面中定义一个表格。但是<table>元素并不能单独地定义一个完整的表格，而是必须要在其中使用<tr>元素以及其他的元素，才能共同构成一个完整的表格。语法结构如下所示。

```
<table>……</table>
```

下面是一个使用<table>元素的实例。其代码如下所示。

例程 7-1　table.html

```
<body>
<table>
<tr>
<th>表格内容</th>
</tr>
<tr>
<th>表格内容</th>
</tr>
```

```
07    </table>
08    </body>
```

该实例中，在 03 和 07 行之间的<table>元素中，使用了<tr>和<th>元素。其运行后的显示效果，如图 7-2 所示。

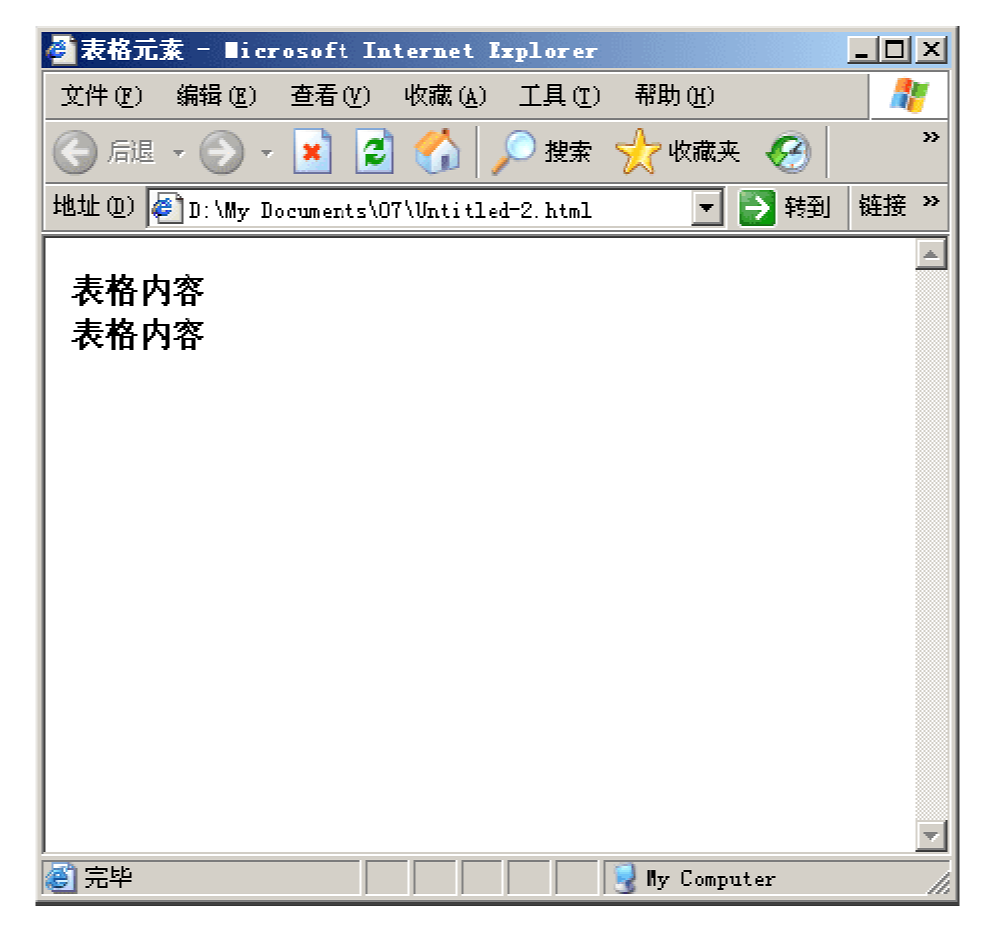


图 7-2　使用<table>元素的显示效果

<table>元素中可以使用的所有属性如表 7-1 所示。

表 7-1　<table>元素的所有属性

属 性 名 称	写　　法
文本显示方向属性	dir
指定语言属性	lang
类属性	class
定义级联样式属性	style
对齐属性	align
标题属性	title
取消自动换行属性	nowrap
标记属性	id
背景颜色属性	bgcolor
背景图片属性	background
边框属性	border
高度属性	height
宽度属性	width
左右边距属性	hspace
上下边距属性	vspace
边框颜色属性	bordercolor

(续表)

属性名称	写　法
暗边框属性	bordercolordack
亮边框属性	bordercolorright
表格边框属性	frame
表格单元格边框属性	rules
表格单元格间距属性	cellspacing
单元格补白属性	cellspadding
列数属性	cols
扩展属性	summary

7.1.2　行<tr>

表格元素<table>用来在页面中定义一个表格。但是<table>元素并不能单独地定义一个完整的表格，而是必须要在其中使用<tr>元素及其他的元素，这样才能共同构成一个完整的表格。其语法结构如下所示。

```
<table>
    <tr>……</tr>
    </table>
```

<tr>元素必须在<table>元素中才能使用。

下面是一个使用<tr>元素的实例。其代码如下所示。

例程 7-2　tr.html

```
01  <body>
02  <h3>这是一个表格：</h3>
03  <table>
04  <tr>
        <td>第一行中的第一个单元格</td>
        <td>第一行中的第二个单元格</td>
05  </tr>
06  <tr>
        <td>这是表格的第二行</td>
        <td>第二行中的第二个单元格</td>
07  </tr>
08  </table>
09  </body>
```

该实例中，04 到 07 行使用<tr>元素定义了两行内容。其中在<tr>元素中定义了<td>元素。其运行后的显示效果，如图 7-3 所示。

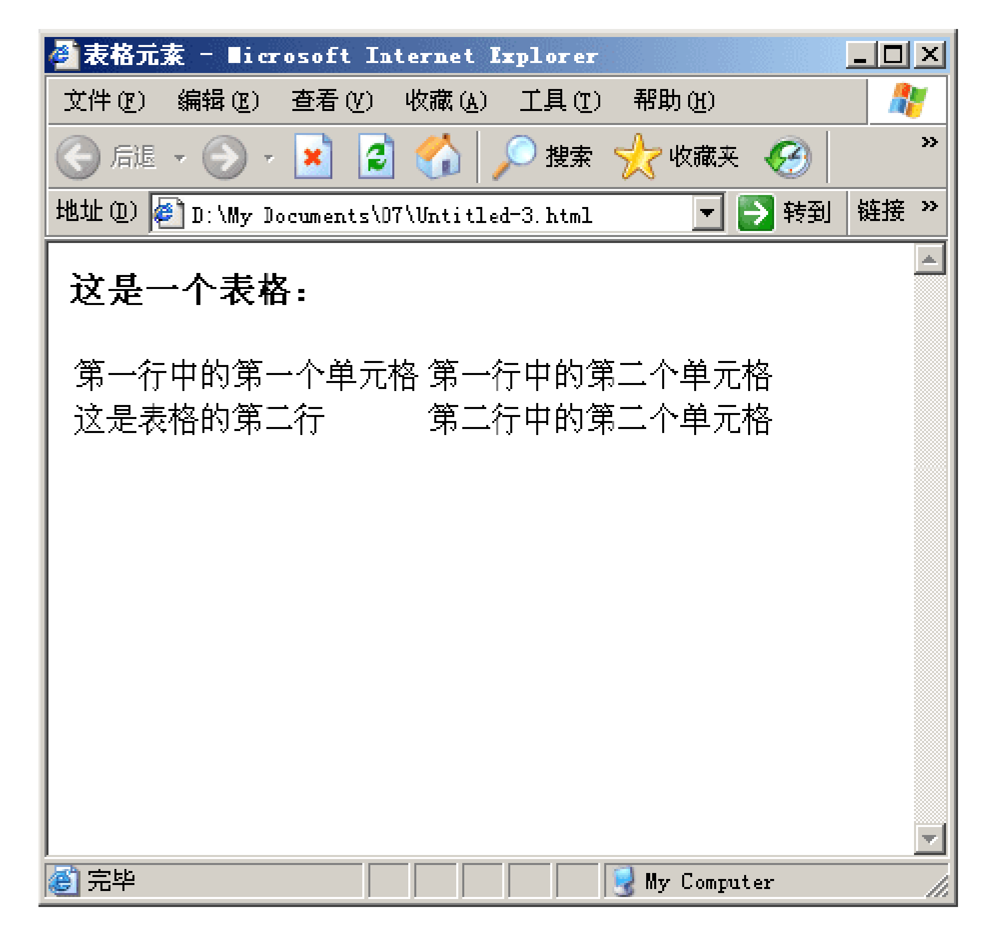


图 7-3　使用<tr>元素的显示效果

<tr>元素中可以使用的所有属性如表 7-2 所示。

表 7-2　<tr>元素的所有属性

属 性 名 称	写　　法
文本显示方向属性	dir
指定语言属性	lang
类属性	class
定义级联样式属性	style
对齐属性	align
标题属性	title
取消自动换行属性	nowrap
标记属性	id
背景颜色属性	bgcolor
垂直对齐属性	valign
边框颜色属性	bordercolor
暗边框属性	bordercolordack
亮边框属性	bordercolorright

7.1.3　单元格<td>

单元格元素<td>用来定义表格的单元格。单元格的含义就是表格中最小的表格单元，也就是几行几列的表格中的一个单独的格子。同样<td>元素不能单独使用，也要与<table>元素、<tr>元素等共同构成一个完整的表格。其语法结构如下所示。

```
<table>
<tr>
<td>……</td>
```

```
    </tr>
</table>
```

<td>元素必须在<table>元素的<tr>元素中才能使用。

下面是一个使用<td>元素的实例。其代码如下所示。

例程 7-3　td.html

```
01  <body>
02  <h3>这是一个表格：</h3>
03  <table>
        <tr>
        <td>第一个单元格</td>
        <td>第二个单元格</td>
        </tr>
        <tr>
        <td>这是表格的第二行</td>
        <td>第二行中的第二个单元格</td>
        </tr>
04  </table>
05  </body>
```

该实例中，03 到 04 行之间的<tr>元素中，使用<td>元素定义了 4 个单元格。其运行后的显示效果，如图 7-4 所示。

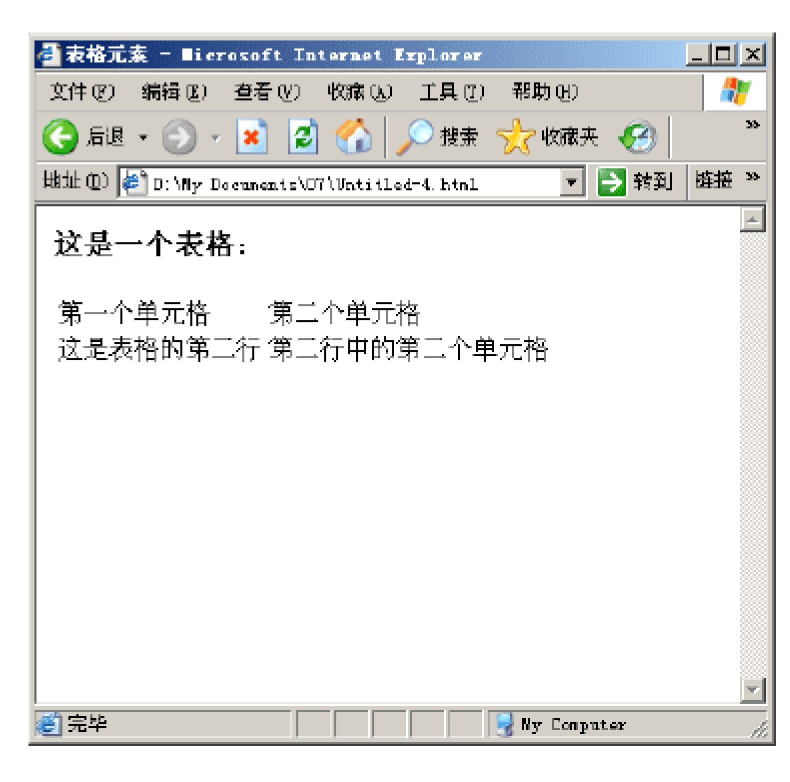


图 7-4　使用<td>元素的显示效果

7.2　<table>元素的属性

通过 7.1 节的学习，读者已经对表格有了初步的认识，在这一节中，本书将详细讲解关于<table>元素的各种相关属性。

7.2.1　边框属性 border

7.1 节讲解表格元素的时候，大家已经注意到页面中看不到边框。在制作过程中，用边框

属性 border 来定义表格元素的边框宽度。如果单元格中没有定义其他的边框属性，则单元格将使用 1 像素的边框属性值。语法结构如下所示。

```
<table border="数字值">……</table>
```

其中数字值的实际单位是像素。下面是在<table>元素中使用 border 属性的实例。其代码如下所示。

例程 7-4 table-border.html

```
01  <body>
02  <h3>这是第一个表格：</h3>
03  <table border="3">
    <tr>
    <td>文学艺术</td><td>天文地理</td><td>政治历史</td>
    </tr>
04  </table>
      <br />
05  <h3>这是第二表格：</h3>
06  <table border="9">
     <tr>
      <td>自然科学</td><td>社会科学</td><td>哲学著作</td>
     </tr>
07  </table>
08  </body>
```

在该实例中，03 行和 06 行定义了两个边框宽度不同的图片，其中一个为 2 像素宽，一个为 10 像素宽，两个表格之间用
元素分隔。其运行后的显示效果，如图 7-5 所示。

从图 7-5 可以看出，此时虽然没有定义单元格的边框属性，但是单元格会显示 1 像素的边框，同时当更改<table>元素的边框属性值时，单元格的边框并不发生变化。

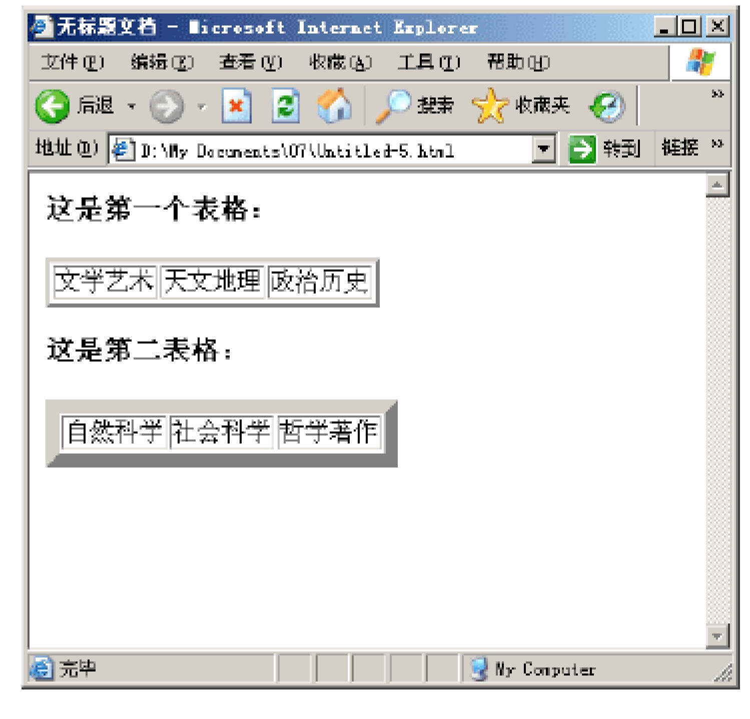


图 7-5 边框宽度不同的表格的表现效果

7.2.2 水平对齐属性 align

在制作页面的过程中，用水平对齐属性 align 来定义表格的对齐方式。根据实际情况来调整表格在页面中的位置，align 语法结构如下所示。

```
<table align="属性值">……</table>
```

align 属性常用的取值有 left、right 和 center。下面是一个在<table>元素中使用 align 属性的实例。其代码如下所示。

例程 7-5 table-align.html

```
<body>
<table align="center" width=400>
<caption>好友通讯录</caption>
```

```
        <tr>
            <th>姓名</th>
            <th>地址</th>
            <th>电话</th>
        </tr>
        <tr>
            <td>李明</td>
            <td>山西省大同市</td>
            <td>0000-12345678</td>
        </tr>
        <tr>
            <td>张三</td>
            <td>河北石家庄市</td>
            <td>0000-12345678</td>
        </tr>
        <tr>
            <td>王五</td>
            <td>河南郑州市</td>
            <td>0000-12345678</td>
        </tr>
04  </table>
05  </body>
```

在该实例中，03 和 04 行的内容是表格的主体，并定义了表格居中显示。其运行后的显示效果，如图 7-6 所示。

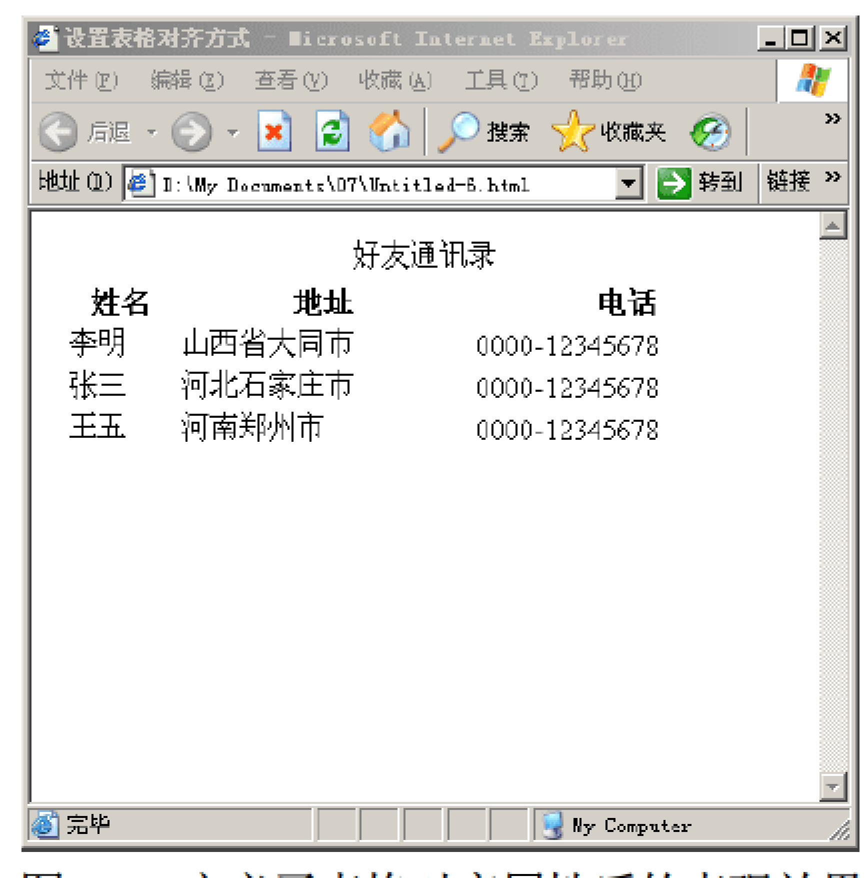


图 7-6　定义了表格对齐属性后的表现效果

7.2.3　高度属性 height

页面制作的时候，还需要调整表格的高度。在 HTML 中，用高度属性 height 来定义表格的高度。如果单元格中没有定义自身的高度属性，则单元格将根据自身含有的内容自动调整高度，适应<table>元素的高度。语法结构如下所示。

```
<table height="数字值">……</table>
```

其中数字值的实际单位是像素。下面是在<table>元素中使用 height 属性的实例。其代码如下所示。

例程 7-6　table-height.html

```
01  <body>
02  <table border="1" width="100" height="100">
      <tr>
        <td>唐诗三百首</td>
      </tr>
      <tr>
        <td>纸上得来终觉浅，</td>
      </tr>
```

```
        <tr>
          <td>纸上得来终觉浅，绝知此事要躬行。</td>
        </tr>
03    </table>
04    </body>
```

在该实例中，02 行中定义了<table>元素的宽度为 100 像素，高度为 200 像素，(宽度属性在 7.2.4 节学习)同时在单元格中分别使用了不同长度的内容。其运行后的显示效果，如图 7-7 所示。

从图 7-7 中可以看到随着内容的增多，因为宽度不变，高度自动做了调整。



图 7-7　定义了表格高度后的表现效果

7.2.4　宽度属性 width

表格的宽度也是页面制作的一个重要内容，在 HTML 中，用宽度属性 width 来定义表格的宽度。如果单元格中没有定义自身的宽度属性，则单元格会根据自身含有的内容自动调整宽度来适应<table>元素的宽度。语法结构如下所示。

```
<table width="数字值">……</table>
```

其中数字值的实际单位是像素。下面是在<table>元素中使用 width 属性的实例。其代码如下所示。

例程 7-7　table-width.html

```
01    <body>
02    <table border="2" width="300">
      <tr>
        <td>无题</td><td>唐代诗人李商隐作</td><td>身无彩凤双飞翼，心有灵犀一点通。</td>
      </tr>
03    </table>
04    </body>
```

在该实例中，02 行中定义了<table>元素的宽度为 300 像素，同时在 3 个单元格中分别添加了不同长度的内容。其运行后的显示效果如图 7-8 所示。



图 7-8　定义了表格宽度后的表现效果

7.2.5　边框颜色属性 bordercolor

为了使表格在页面美观，用边框颜色属性 bordercolor 来定义表格边框的颜色。语法结构如下所示。

注意

<table>元素中定义的边框颜色，将会影响到单元格中的边框颜色，如果单元格中没有定义自身的边框颜色，单元格中将使用<table>元素中定义的边框颜色。

```
<table border="数字值" bordercolor="颜色值">……</table>
```

当使用 bordercolor 属性时，要定义相应的 border 属性，否则 bordercolor 属性将没有表现效果。下面是一个在<table>元素中使用 bordercolor 属性的实例。其代码如下所示。

例程 7-8 table-bordercolor.html

```
01  <body>
02  <table border="8" width="400" height="400" bordercolor="#00FF00">
    <tr>
     <td>学科科目</td><td>学科学时</td>
    </tr>
     <tr>
      <td>自然科学</td><td>60 学时</td>
     </tr>
     <tr>
      <td>社会科学</td><td>60 学时</td>
     </tr>
03  </table>
04  </body>
```

在该实例中，02 行中定义了<table>元素的宽度为 400 像素，高度为 400 像素，为了明显地看到边框颜色，这里定义边框宽度为 8 个像素，颜色为绿色。其代码运行后的显示效果，如图 7-9 所示。

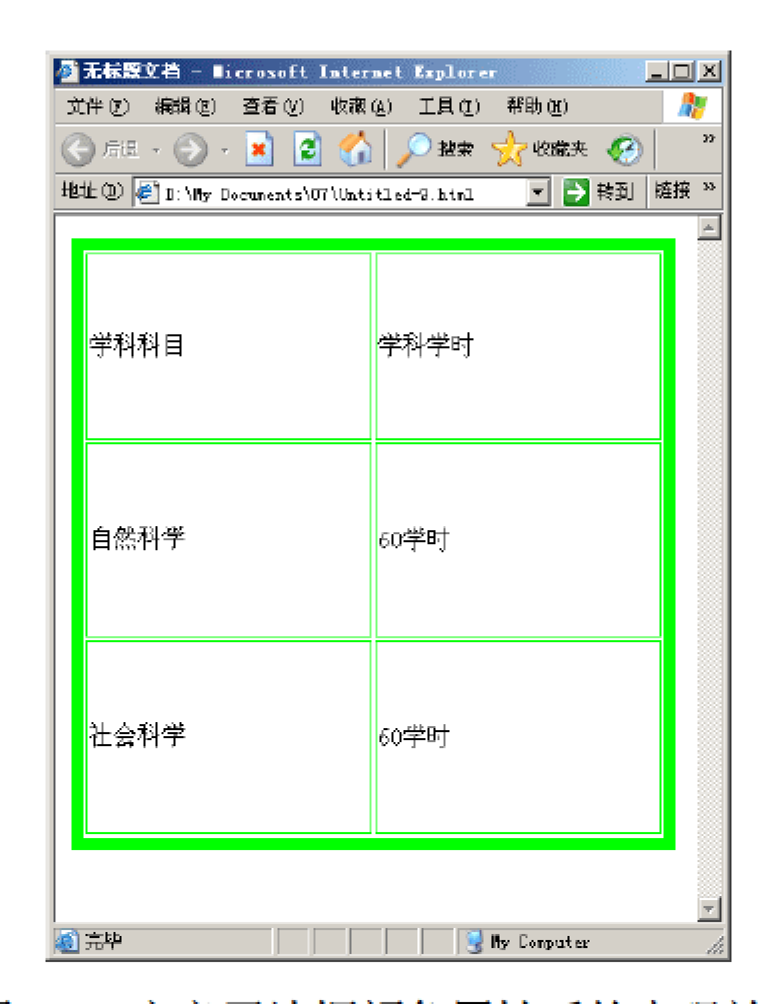


图 7-9　定义了边框颜色属性后的表现效果

7.2.6　边框暗边线属性 bordercolordark

这里先来了解一下什么是边框暗边线属性，在<table>元素中边框暗边线是指表格的右侧和底部的边线。用边框暗边线属性 bordercolordark 来定义表格边框暗边线的颜色。语法结构如下所示。

```
<table border="数字值" bordercolordark=" 颜色值">……
</table>
```

在这个语法中要注意，当使用 bordercolordark 属性时，要定义相应的 border 属性，否则 bordercolordark 属性将没有表现效果。下面是一个在<table>元素中使用 bordercolordark 属性的实例。其代码如下所示。

例程 7-9　table-bordercolordark.html

```
01  <body>
02  <table border="8" width="400" height="400" bordercolor="#99FF99" bordercolordark="#000000">
    <tr>
      <td>自然科学</td><td>人文科学</td>
    </tr>
     <tr>
      <td>物理化学等知识</td><td>社会哲学等知识</td>
     </tr>
03  </table>
04  </body>
```

在该实例中，02 行中定义了<table>元素的宽度为 400 像素，高度为 400 像素，边框宽度为 8 个像素，边框颜色为浅绿色，边框暗边线的颜色为黑色。其代码运行后的显示效果，如图 7-10 所示。

从图 7-10 可以看出，<table>元素使用边框暗边线颜色属性后，表格的右侧和底部的边线的颜色发生了变化。

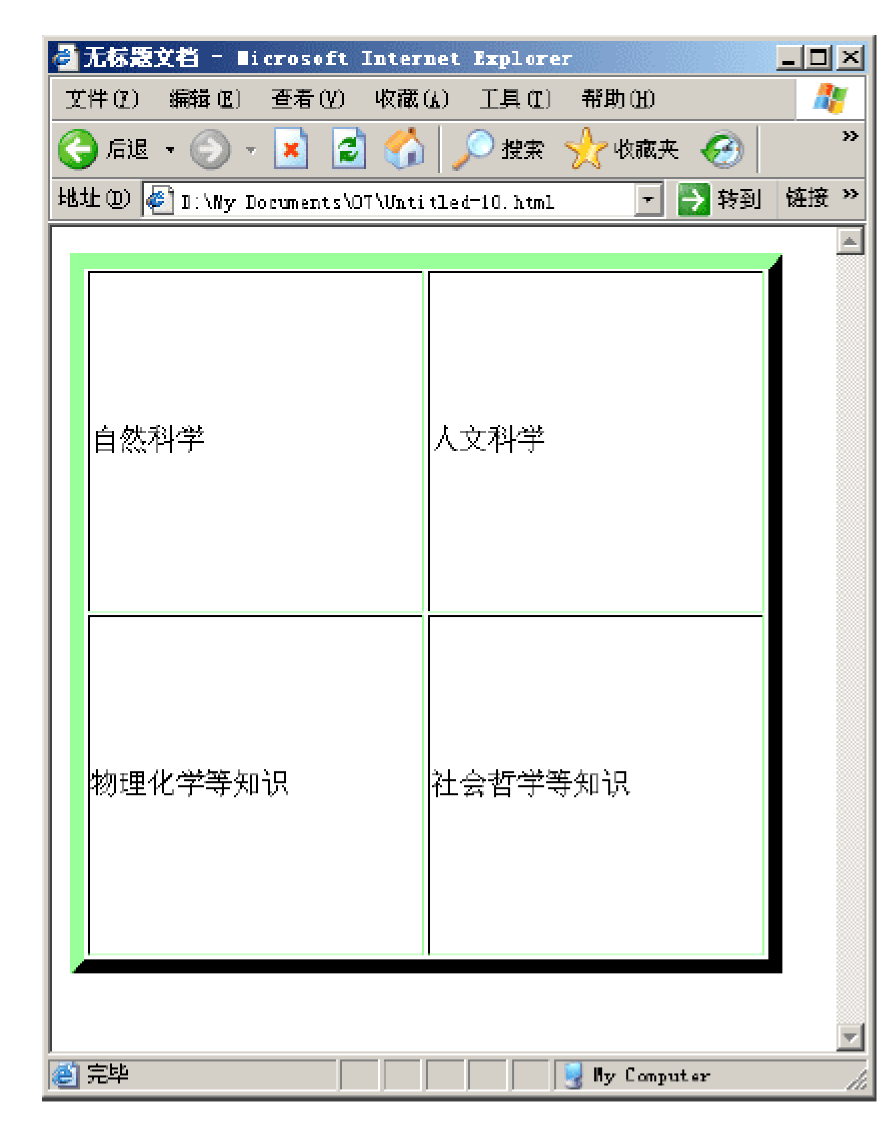


图 7-10　定义了边框暗边线颜色属性后的效果

7.2.7　边框亮边线属性 bordercolorlight

<table>元素中边框亮边线是指表格的左侧和顶部的边线。用边框亮边线属性 bordercolorlight 来定义表格边框亮边线的颜色。其语法结构如下所示。

```
<table border="数字值" bordercolorlight=" 颜色值">……</table>
```

当使用 bordercolorlight 属性时，要定义相应的 border 属性，否则 bordercolorlight 属性将没有表现效果。下面是一个在<table>元素中使用 bordercolorlight 属性的实例。其代码如下所示。

例程 7-10　table-bordercolorlight.html

```
01  <body>
02  <table border="8" width="400" height="400" bordercolor="#999999" bordercolorlight="#00FF99">
    <tr>
     <td>初唐四杰</td><td>苏门四学士</td>
    </tr>
    <tr>
     <td>王勃、杨炯、卢照邻、骆宾王。</td><td>黄庭坚、秦观、晁补之和张耒的并称</td>
    </tr>
03  </table>
04  </body>
```

在上面的实例中，02 行中定义了<table>元素的宽度为 400 像素，高度为 400 像素，边框

宽度为 5 个像素，边框颜色为浅灰色，边框亮边线的颜色为浅绿色。其代码运行后的显示效果，如图 7-11 所示。

从图 7-11 可以看出，<table>元素使用边框亮边线颜色属性后，表格的左侧和顶部的边线颜色发生了变化。

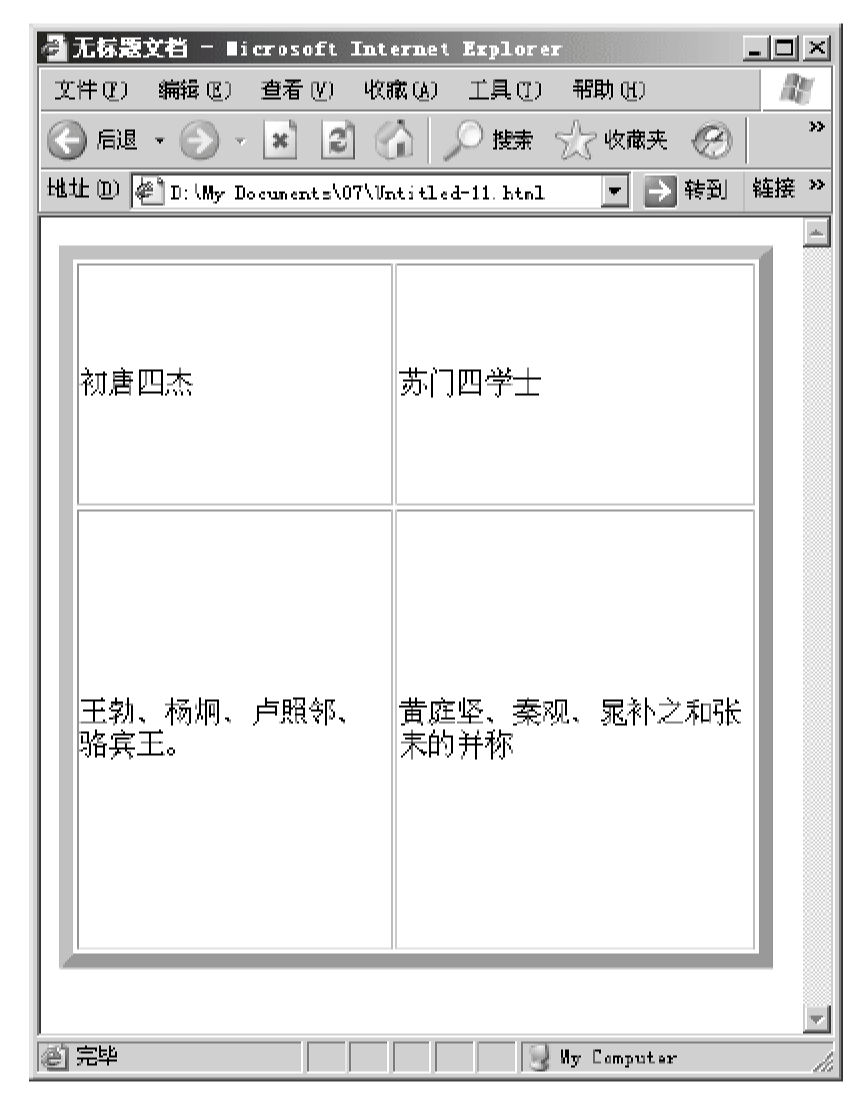


图 7-11　定义了边框亮边线颜色属性后的效果

7.2.8　背景颜色属性 bgcolor

为了突出显示表格，还可以为表格设置与页面不同的背景。通常用背景颜色属性 bgcolor 来定义表格的背景颜色。其语法结构如下所示。

```
<table bgcolor="颜色值">其他表格</table>
```

其中颜色值既可以使用颜色名称，也可以使用 16 进制的颜色值。下面是一个在<table>元素中使用 bgcolor 属性的实例。其代码如下所示。

例程 7-11　table-bgcolor.html

```
<body>
<table bgcolor="#FFFF00" border=4 bordercolor="#990000">
<caption>通讯录</caption>
  <tr>
       <th>姓名</th>
       <th>地址</th>
       <th>固定电话</th>
       <th>电话</th>
    </tr>
    <tr>
       <td>张三</td>
       <td>广西南宁</td>
       <td>0000-123456677</td>
       <td>15088946572</td>
     </tr>
     <tr>
       <td>李四</td>
       <td>陕西西安</td>
       <td>0000-32145687</td>
       <td>13965478635</td>
     </tr>
     <tr>
       <td>王五</td>
       <td>天津市南海路</td>
       <td>000-123654789</td>
       <td>13545700506</td>
      </tr>
</table>
</body>
```

本例程中 02 行设置<table>元素的宽度和高度默认，背景颜色为浅橙色。其代码运行后的显示效果，如图 7-12 所示。

从图 7-12 可以看出，<table>元素的背景不但显示在表格之中，同时也会显示在默认的部分边框之中。



图 7-12　定义了表格背景颜色属性后的表现效果

7.2.9　背景图片属性 background

为了让表格更加绚丽，除了可以为表格设置背景色之外，还可以设置一个背景图像。通常用背景图片属性 background 来定义表格使用的背景图片。如果单元格中没有定义自身的背景颜色属性或者图片属性，则单元格将显示<table>元素中定义的背景图片。其语法结构如下所示。

```
<table background="图片路径">……</table>
```

其中图片路径既可以使用相对路径，也可以使用绝对路径。下面是一个在<table>元素中使用 background 属性的实例。其代码如下所示。

例程 7-12　table-background.html

```
01	<body>
02	<table border="1" width="300" height="300" background="http://www.w3.org/Icons/w3c_main">
	<tr>
	  <td>表格内容 1</td><td>表格内容 1</td>
	</tr>
	<tr>
	<td>表格内容 2 增加内容</td><td>表格内容 2 增加内容</td>
	</tr>
03	</table>
04	</body>
```

在该实例中，02 行中定义了<table>元素的宽度为 300 像素，高度为 300 像素，背景图片为 W3C 网站的 logo。其代码运行后的显示效果，如图 7-13 所示。

从图 7-13 可以看出，<table>元素使用的背景图片将按照从左至右、从上到下的顺序重复排满整个表格。

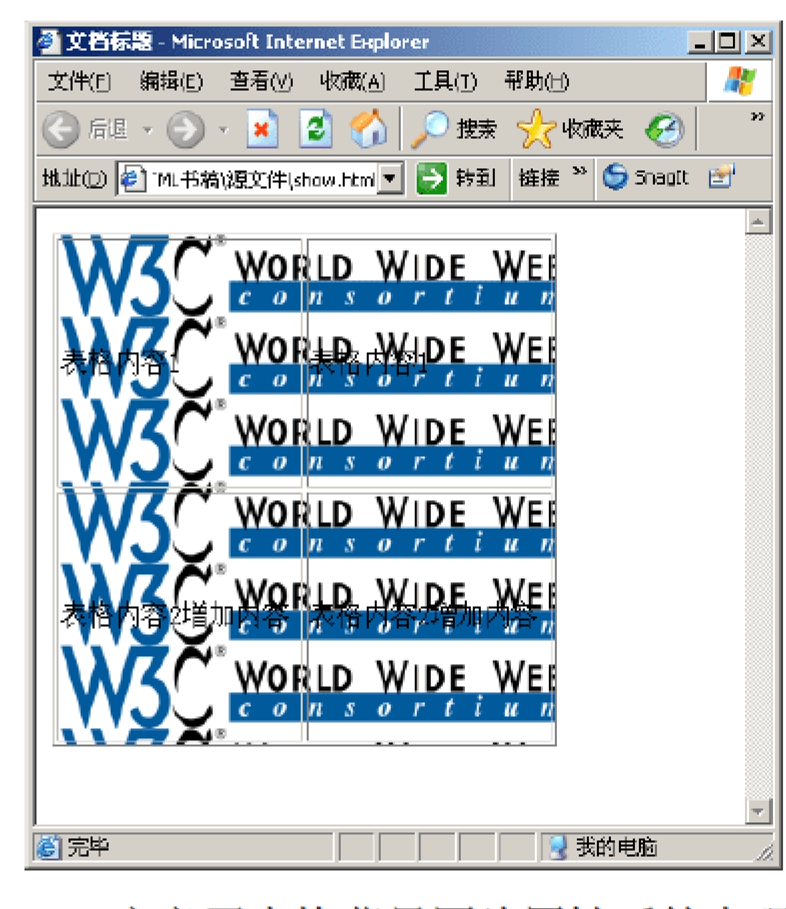


图 7-13　定义了表格背景图片属性后的表现效果

7.2.10　单元格间距属性 cellspacing

设计网页时，有时候需要把各个单元格独

立。在 HTML 中，通常用单元格间距属性 cellspacing 来定义各个单元格之间的间距。使用 cellspacing 属性定义单元格之间间距的同时，也使单元格和<table>元素之间分隔了一定距离。其语法结构如下所示。

```
<table border="数字值" cellspacing="数字值">……</table>
```

其中数字值实际的单位是像素。下面是一个使用 cellspacing 属性的实例。其代码如下所示。

例程 7-13 table-cellspacing.html

```
01  <body>
02  <table width=560 border=4 bordercolor="#000000" cellspacing=15>
03  <caption>课程表</caption>
            <tr>
                <th>星期一</th>
                <th>星期二</th>
                <th>星期三</th>
                <th>星期四</th>
              <th>星期五</th>
            </tr>
            <tr>
                <td>语文</td>
                <td>数学</td>
                <td>英语</td>
                <td>数学</td>
              <td>语文</td>
            </tr>
            <tr>
                <td>数学</td>
                <td>英语</td>
                <td>语文</td>
                <td>英语</td>
              <td>英语</td>
            </tr>
            <tr>
                <td>历史</td>
                <td>地理</td>
                <td>政治</td>
                <td>历史</td>
              <td>物理</td>
            </tr>
04  </table>
05  </body>
```

在该实例中，02 行中定义了<table>元素的宽度为 560 像素，边框宽度为 4 个像素，单元格间距为 15。其代码运行后，可以看到各个单元格是独立的，并且和表格之间有一定的距离。显示效果如图 7-14 所示。

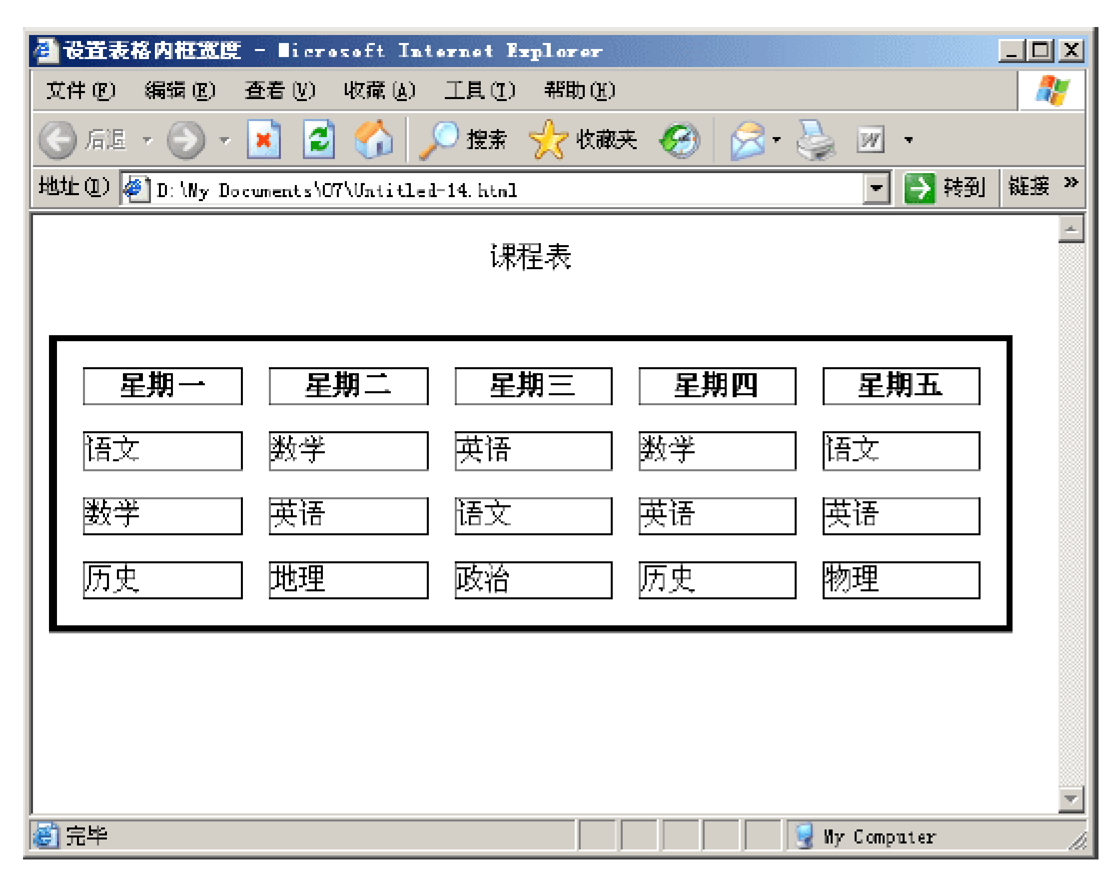


图 7-14　定义了单元格间距属性后的效果

7.2.11　单元格补白属性 cellspadding

在默认情况下，表格内的文字会紧贴着表格的边框，这样看上去非常拥挤。可以使用单元格补白属性 cellspadding 来调整各个单元格与其中的内容之间的间距。语法结构如下所示。

```
<table border="数字值" cellspadding ="数字值">……</table>
```

其中数字值实际的单位是像素。下面是一个使用 cellspadding 属性的实例。其代码如下所示。

例程 7-14　table-cellspadding.html

```
<body>
<table border=4 bordercolor="#FF00FF" cellspacing=6 cellpadding=20>
<caption>课程表</caption>
        <tr>
            <th>星期一</th>
            <th>星期二</th>
            <th>星期三</th>
            <th>星期四</th>
          <th>星期五</th>
        </tr>
        <tr>
            <td>语文</td>
            <td>数学</td>
            <td>英语</td>
            <td>数学</td>
          <td>语文</td>
        </tr>
        <tr>
            <td>数学</td>
            <td>英语</td>
            <td>语文</td>
            <td>英语</td>
          <td>英语</td>
        </tr>
        <tr>
```

```
                <td>历史</td>
                <td>地理</td>
                <td>政治</td>
                <td>历史</td>
              <td>物理</td>
           </tr>
04    </table>
05    </body>
```

在该实例中，02 行中定义了<table>元素边框宽度为 2 个像素，单元格补白属性值为 20。其代码运行后的显示效果，如图 7-15 所示。

从图 7-15 可以看出，定义单元格补白属性后，相应的单元格的内容和单元格之间的距离发生了变化。



图 7-15　定义了单元格补白属性后的效果

7.2.12　表格单元格边框属性 rules

有时候需要根据页面的实际需要定义单元格之间的边框，在 HTML 中通常用表格单元格边框属性 rules 来在表格中定义单元格之间的边框，也可以定义列、行或者列组等元素的边框。语法结构如下所示。

```
<table ruls="属性值">……</table>
```

其中 rules 属性值的各个具体含义如表 7-3 所示。

表 7-3　align 属性的取值及含义

属　性　值	代表的含义
all	使用所有边框
groups	右行或列组中使用边框
rows	在行之间使用边框
cols	在列之间使用边框

7.2.13　表格边框属性 frame

表格边框属性 frame 用来定义表格边框的显示方式，通过 frame 属性可以定义表格的 4 个边框是否显示，或者定义某一侧的边框显示与否。语法结构如下所示。

```
<table frame="属性值">其他内容</table>
```

其中 frame 属性值的具体含义如表 7-4 所示。

表 7-4　frame 属性的取值及含义

属　性　值	代表的含义
box	使用所有边框(默认值)
void	删除所有边框
above	显示上边框
blow	显示下边框
lhs	显示左边框
rhs	显示右边框
nsides	显示顶部和底部的边框
vsides	显示左侧和右侧的边框

下面是一个使用 frame 属性的实例。其代码如下所示。

例程 7-15　table-frame.html

```
01  <body>
02  <table border="4" width="400" height="200" frame="lhs">
03  <caption>课程表</caption>
        <tr>
            <th>星期一</th>
            <th>星期二</th>
            <th>星期三</th>
            <th>星期四</th>
          <th>星期五</th>
        </tr>
        <tr>
            <td>语文</td>
            <td>数学</td>
            <td>英语</td>
            <td>数学</td>
          <td>语文</td>
        </tr>
        <tr>
            <td>数学</td>
            <td>英语</td>
            <td>语文</td>
            <td>英语</td>
          <td>英语</td>
        </tr>
04  </table>
05  </body>
```

在该实例中，02 行中使用了 frame 属性的 lhs，定义在表格中只显示其左边框。其代码运行后的显示效果，如图 7-16 所示。



图 7-16　定义了表格 frame 属性后的效果



7.3 <tr>元素的属性

前面学习了<table>元素的属性，接下来，将学习<tr>元素的相关属性。

7.3.1　水平对齐属性 align

在 HTML 中通常用水平对齐属性 align 来定义行所包含的单元格内容的水平对齐方式。语法结构如下所示。

```
<table>
<tr align="属性值">其他内容</tr>
    </table>
```

align 属性的常见齐方式包含 3 种，分别为 left、center 和 right。下面是一个在<tr>元素中使用 align 属性的实例。其代码如下所示。

例程 7-16　tr-align.html

```
<body>
<table border=4 bordercolor="#0000FF" >
<caption>我国著名科学家——沈括</caption>
<tr align="center">
<td>朝代</td>
<td>宋代</td>
</tr>
<tr align="right">
<td>著作</td>
<td>梦溪笔谈</td>
</tr>
<tr align="left" height=90>
```

```
    <td>简介</td>
    <td> 沈括是我国北宋著名的科学家，他的梦溪笔谈是我国最早的科学著作之一</td>
08  </tr>
09  </table>
10  </body>
```

在该实例中，02 行中定义了<table>元素的边框宽度为 4 个像素，边框颜色为蓝色。04 行中定义了对齐属性值为 center，06 行定义了对齐属性值为右对齐，07 行定义了对齐属性值为左对齐，通过定义<tr>元素中的 align 属性，实现了某一行内容的居中。其代码运行后的显示效果，如图 7-17 所示。

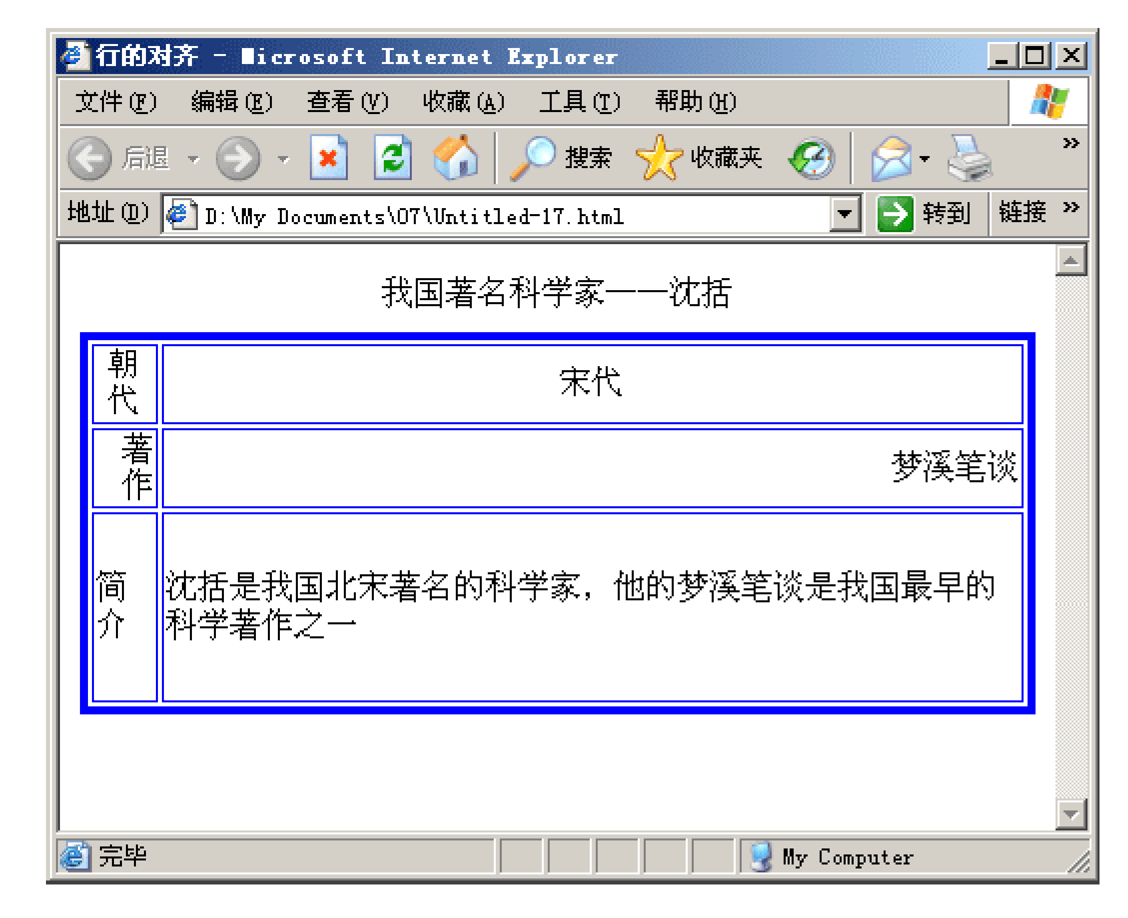


图 7-17　在<tr>元素中定义 align 属性后的表现效果

7.3.2　垂直对齐属性 valign

在制作表格的时候，也可以把单独的一行设置成特殊对齐方式。在 HTML 中通常用垂直对齐属性 valign 来定义行所包含的单元格内容的垂直对齐方式。语法结构如下所示。

```
<table>
<tr valign="属性值">其他部分</tr>
</table>
```

valign 属性的取值包含 3 种，分别为 top、middle 和 bottom。下面是一个在<tr>元素中使用 valign 属性的实例。其代码如下所示。

例程 7-17　tr-valign.html

```
01  <body>
02  <table border=4 bordercolor="#660000" cellpadding=10>
03  <caption>我国著名科学家——沈括</caption>
04  <tr height=20 valign="top">
    <td>朝代</td>
    <td>宋代</td>
05  </tr>
06  <tr height=50 valign="center">
    <td>著作</td>
    <td>梦溪笔谈</td>
07  </tr>
08  <tr height=90 valign="bottom">
    <td>简介</td>
    <td> 沈括是我国北宋著名的科学家，他的梦溪笔谈是我国最早的科学著作之一</td>
09  </tr>
10  </table>
11  </body>
```

在该实例中，02 行中定义了<table>元素的边框宽度为 2 个像素。04 行中定义了垂直对齐

属性值为 top，文字会在表格上方显示。同样的 06 行中定义了对齐属性值为 center，文字会在表格的中部显示，最后在 08 行中定义了对齐属性值为 bottom，文字会出现在表格底部。其代码运行后的显示效果，如图 7-18 所示。

从图 7-18 可以看出，通过定义<tr>元素中的 valign 属性，实现了某一行内容的顶部对齐。

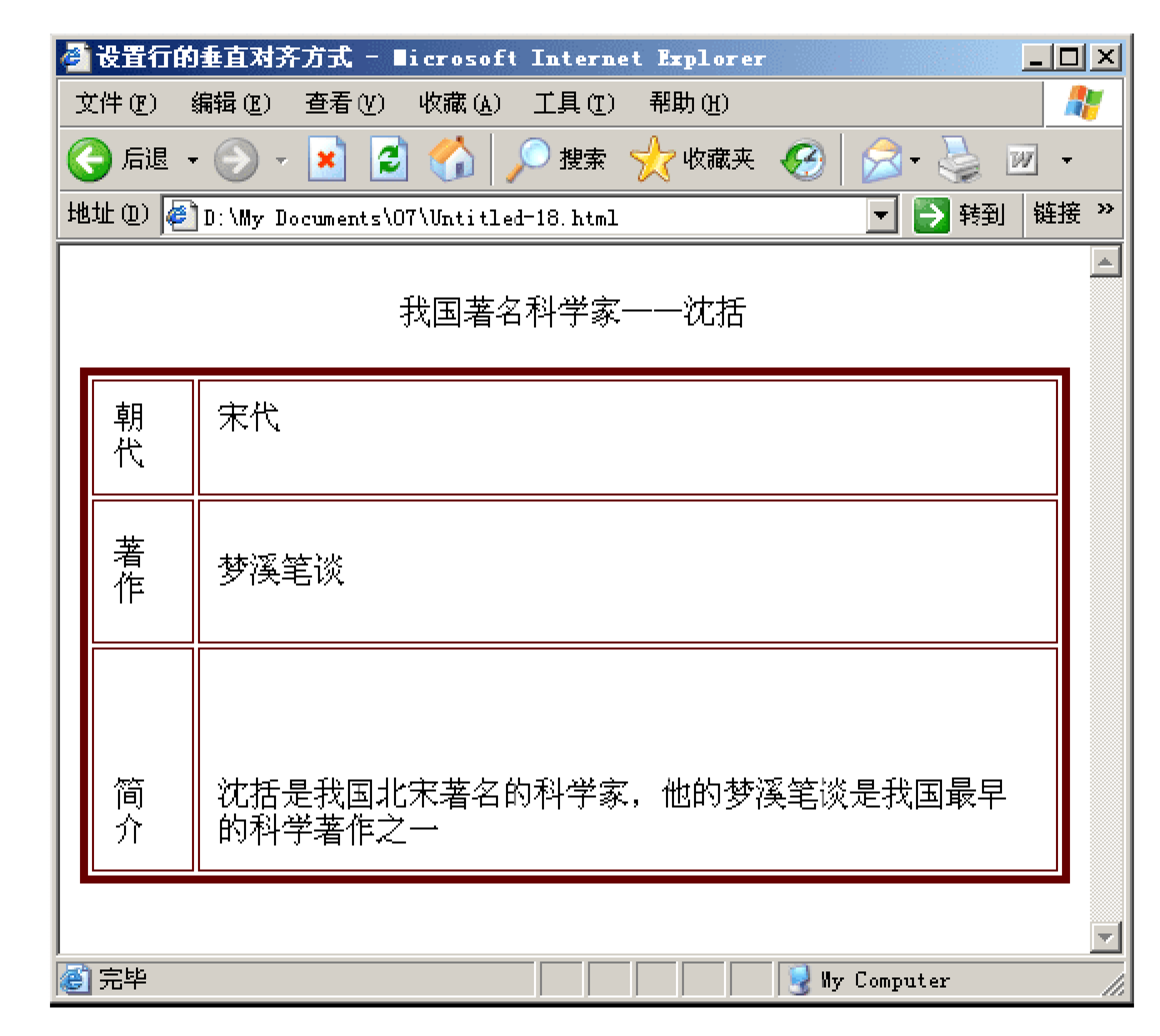


图 7-18　在<tr>元素中定义 valign 属性后的表现效果

7.3.3　背景颜色属性 bgcolor

在 HTML 中通常用背景颜色属性 bgcolor 来定义行的背景颜色。语法结构如下所示。

```
<table>
    <tr bgcolor="颜色值">……</tr>
</table>
```

其中颜色值既可以使用颜色名称，也可以使用 16 进制的颜色值。下面是一个在<tr>元素中使用 bgcolor 属性的实例。其代码如下所示。

例程 7-18　tr-bgcolor.html

```
01  <body>
02  <table width=500 border=1>
03  <caption>甲班部分学员的成绩表</caption>
04  <tr bgcolor="#00FF00">
            <th>姓名</th>
            <th>数学</th>
            <th>语文</th>
            <th>外语</th></tr>
        <tr>
            <td>张三</td>
            <td>88</td>
            <td>87</td>
            <td>91</td></tr>
05  <tr bgcolor="#FFEE00">
            <td>李四</td>
            <td>87</td>
            <td>80</td>
            <td>85</td></tr>
        <tr>
            <td>王五</td>
            <td>80</td>
            <td>78</td>
            <td>95</td></tr>
```

```
06    </table>
07    </body>
```

在该实例中，02 行定义了<table>元素的宽度为 500 像素，边框宽度为 1 个像素。04 行中定义了背景颜色属性值为绿色，05 行中定义了背景颜色属性值为橙色。其代码运行后的显示效果，如图 7-19 所示。

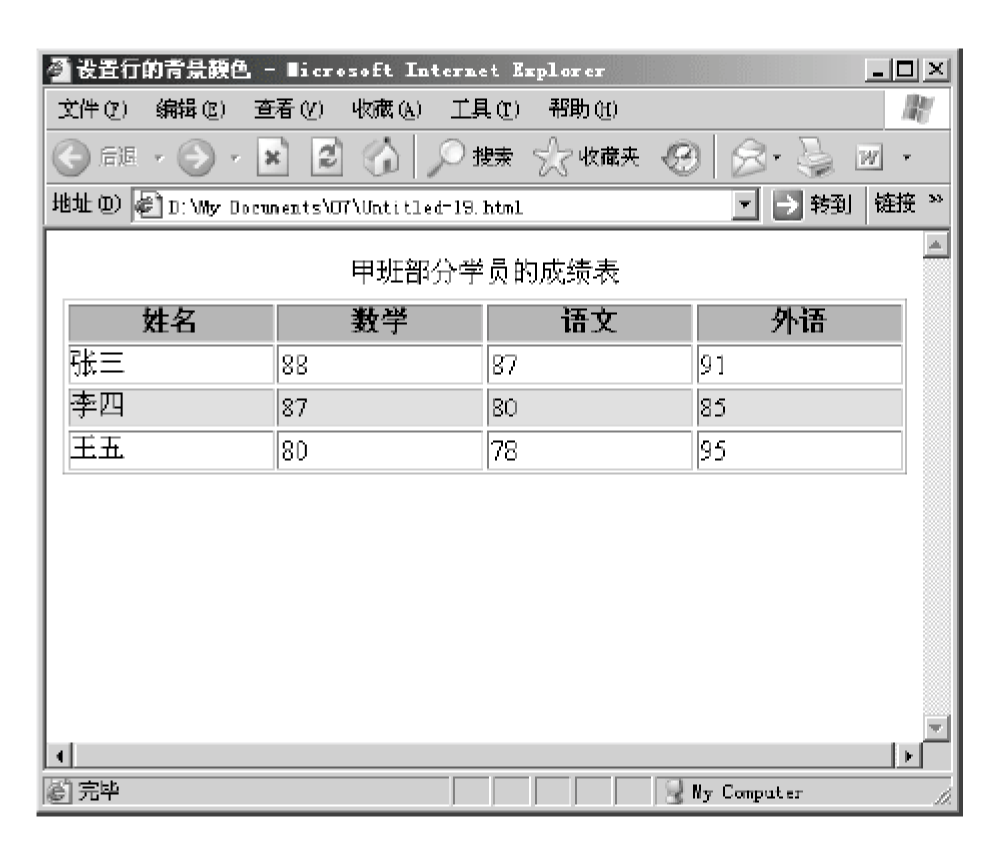


图 7-19　在<tr>元素中定义 bgcolor 属性后的表现效果

7.3.4　边框颜色属性 bordercolor

在 HTML 中通常用边框颜色属性 bordercolor 来定义行内单元格的边框颜色。语法结构如下所示。

```
<table>
    <tr bordercolor ="颜色值">其他内容</tr>
</table>
```

其中颜色值既可以使用颜色名称，也可以使用 16 进制的颜色值。下面是一个在<tr>元素中使用 bordercolor 属性的实例。其代码如下所示。

例程 7-19　tr-bordercolor.html

```
01    <body>
02    <table width="400" height="160" border="6" bordercolor="00CCFF">
03    <caption>甲班部分学员的成绩表</caption>
          <tr>
              <th>姓名</th>
              <th>数学</th>
              <th>语文</th>
              <th>外语</th></tr>
04    <tr bordercolor="#990000">
              <td>张三</td>
              <td>88</td>
              <td>87</td>
              <td>91</td></tr><tr>
              <td>李四</td>
              <td>87</td>
              <td>80</td>
              <td>85</td></tr>
          <tr>
              <td>王五</td>
              <td>80</td>
              <td>78</td>
              <td>95</td></tr>
05    </table>
06    </body>
```

在该实例中，02 行中定义了<table>元素的宽度为 400 像素，高度为 160 像素，边框宽度

为 6 个像素，边框颜色为蓝色。04 行中定义了表格第二行中内边框颜色属性值为绿色。其代码运行后的显示效果，如图 7-20 所示。

从图 7-20 可以看出，<tr>元素中使用 bordercolor 属性，不能改变<table>元素的边框属性。



图 7-20　在<tr>元素中定义 bgcolor 属性后的表现效果

7.3.5　边框暗边线属性 bordercolordark

在 HTML 中通常用边框暗边线属性 bordercolordark 来定义行内单元格的边框暗边线颜色。<tr>元素中的边框暗边线指的是行内单元格的左侧边线和顶部边线。语法结构如下所示。

```
<table>
    <tr bordercolordark ="颜色值">……</tr>
    </table>
```

其中颜色值既可以使用颜色名称，也可以使用 16 进制的颜色值。

下面是一个在<tr>元素中使用 bordercolordark 属性的实例。其代码如下所示。

例程 7-20　tr-bordercolordark.html

```
01  <body>
02  <table border="4" width="500" height="160">
03  <tr bordercolordark="#FF0000"><td>自然科学</td><td>社会科学</td></tr>
04  <tr><td>自然科学是人类征服自然的过程中产生的</td><td>社会科学是人与人相互交流产生的</td>
</tr>
05  <tr><td>自然科学有物理，化学，生物，地理等。</td><td>社会科学有政治、历史和哲学。</td></tr>
06  </table>
07  </body>
```

在该实例中，02 行中定义了第一个<table>元素的边框宽度为 2 个像素，宽度为 500 像素，高度为 160 像素，并在 03 行中定义了表格第一行中边框暗边线属性值为红色。其代码运行后的显示效果如图 7-21 所示。

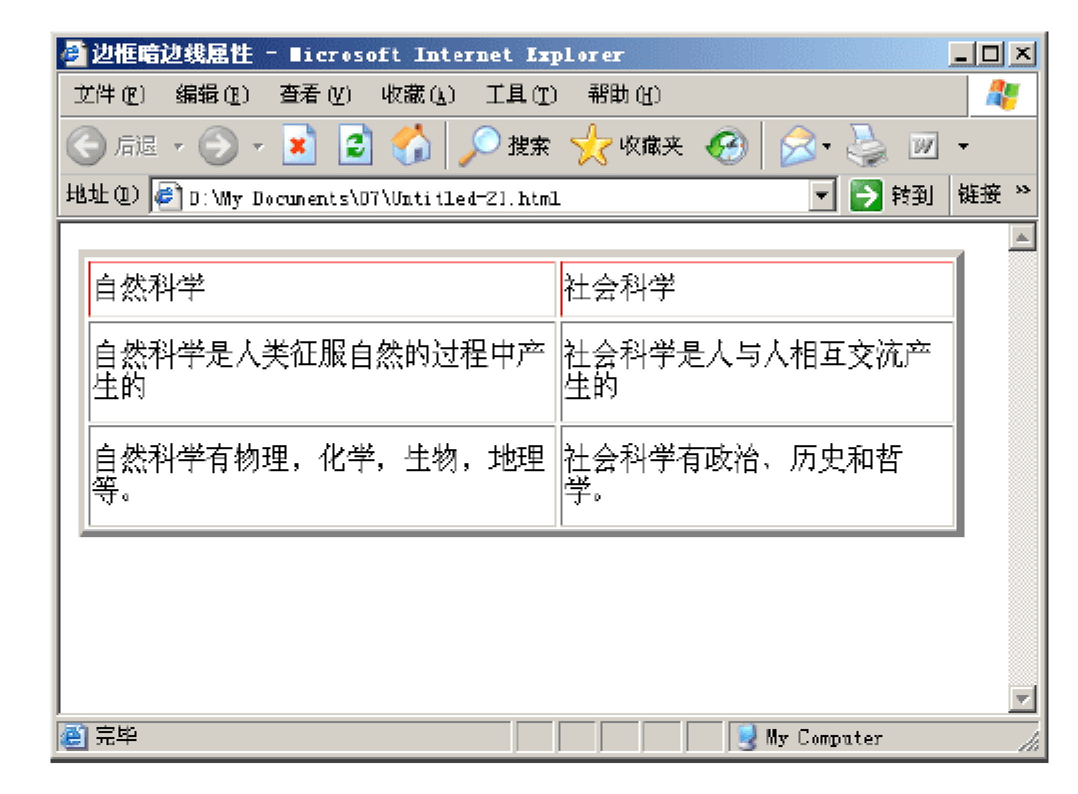


图 7-21　在<tr>元素中定义 bordercolordark 属性后的表现效果

7.3.6　边框亮边线属性 bordercolorlight

同样的也可以用边框暗边线属性 bordercolorlight 来定义行内单元格的边框亮边线颜色。<tr>元素中的边框暗边线指的是行内单元格的右侧边线和底部边线。语法结构如下所示。

```
<table>
```

```
        <tr bordercolorlight ="颜色值">其他表格元素和内容部分</tr>
</table>
```

其中颜色值既可以使用颜色名称，也可以使用 16 进制的颜色值。

下面是一个在<tr>元素中使用 bordercolorlight 属性的实例。其代码如下所示。

例程 7-21　tr-bordercolorlight.html

```
01  <body>
02  <table border="4" width="500" height="160">
03  <tr bordercolorlight="#0000FF"><td>自然科学</td><td>社会科学</td></tr>
04  <tr><td>自然科学是人类征服自然的过程中产生的</td><td>社会科学是人与人相互交流产生的
</td></tr>
05  <tr><td>自然科学有物理，化学，生物，地理等。</td><td>社会科学有政治、历史和哲学。</td></tr>
06  </table>
07  </body>
```

在该实例中，02 行中定义了第一个<table>元素的边框宽度为 2 个像素，宽度为 500 像素，高度为 160 像素，03 行中定义了表格第一行中边框亮边线属性值为蓝色。其代码运行后的显示效果如图 7-22 所示。

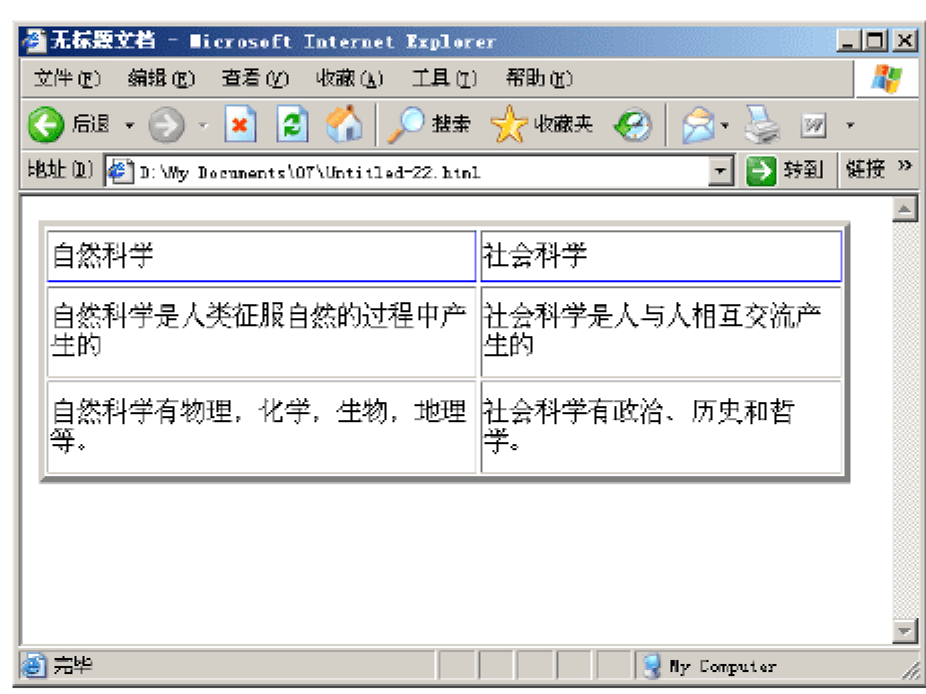


图 7-22　在<tr>元素中定义 bordercolorlight 属性后的表现效果

7.4 <td>元素的属性

本节将学习单元格的标记，单元格的标记与行标记十分相似，学习的时候可以对照 7.3 节来加深对单元格的理解。

7.4.1　宽度属性 width

HTML 中也用宽度属性 width 来定义单元格的宽度。语法结构如下所示。

```
<table>
    <tr>
        <td width="数字值">内容部分</td>
    </tr>
```

```
</table>
```

其中数字值的实际单位是像素。下面是一个在<td>元素中使用 width 属性的实例。其代码如下所示。

例程 7-22 td-width.html

```
01 <body>
02 <table border="4" width="400" height="200">
03 <tr><td width="200">自然科学</td><td>社会科学</td></tr>
04 <tr><td >自然科学关于人类生产发展的科学</td><td >社会科学是人类政治经济发展的科学</td></tr>
05 </table>
06 </body>
```

在这个实例中，02 行定义了<table>元素的边框宽度为 4 个像素，宽度为 400 像素，高度为 220 像素。第一行左侧单元格中定义宽度为 200 像素。其代码运行后的显示效果，如图 7-23 所示。可以看出<td>中定义宽度后，虽然没有定义第二个表格的宽度，但是页面会自动调整宽度使总宽度等于<table>元素的宽度。如果把第二单元格的宽度设置为 300 像素，其代码如下所示。

例程 7-23 td-widthtwo.html

```
<body>
<table border="4" width="400" height="200">
<tr><td width="200">自然科学</td><td width="300">社会科学</td></tr>
<tr><td>自然科学关于人类生产发展的科学</td><td>社会科学是人类政治经济发展的科学</td></tr>
</table><br />
<table border="4" width="400" height="200">
<tr><td>自然科学</td><td>社会科学</td></tr>
<tr><td>自然科学关于人类生产发展的科学</td><td>自然科学关于人类生产发展的科学</td></tr>
</table>
</body>
```

其代码运行后的显示效果，如图 7-24 所示。

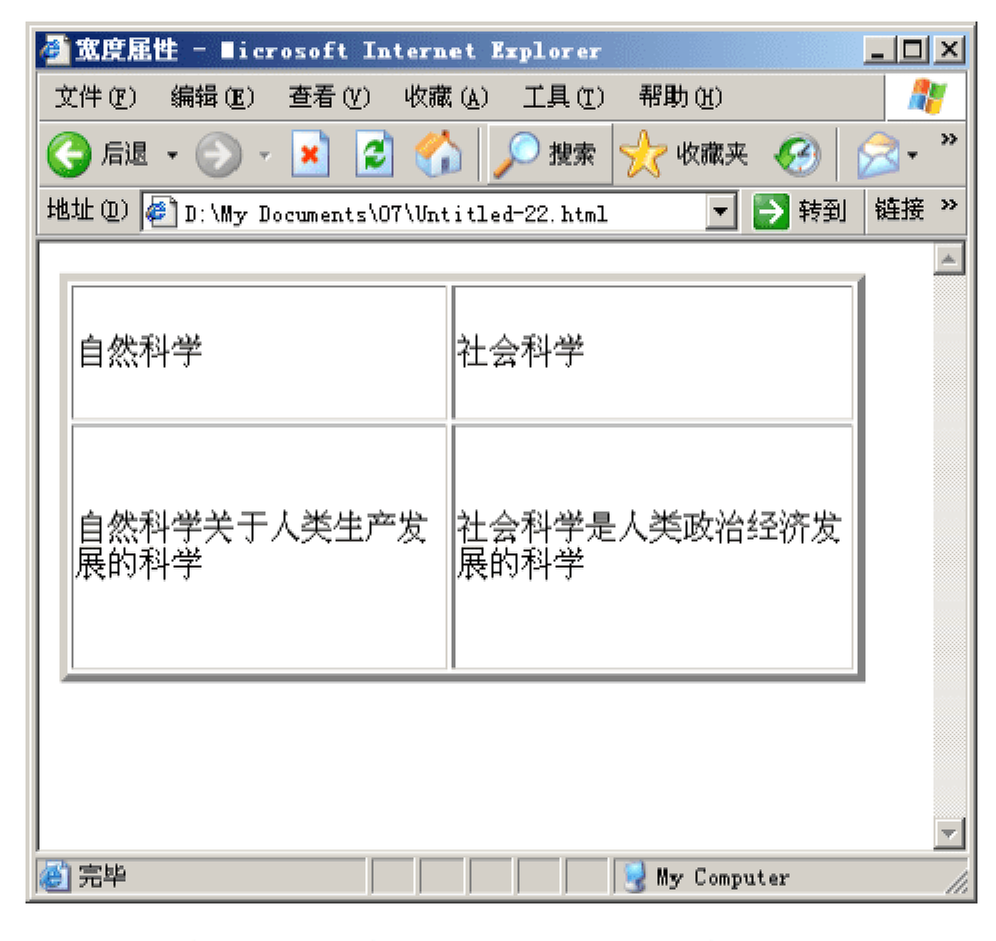


图 7-23 在<td>元素中定义 width 属性后的表现效果

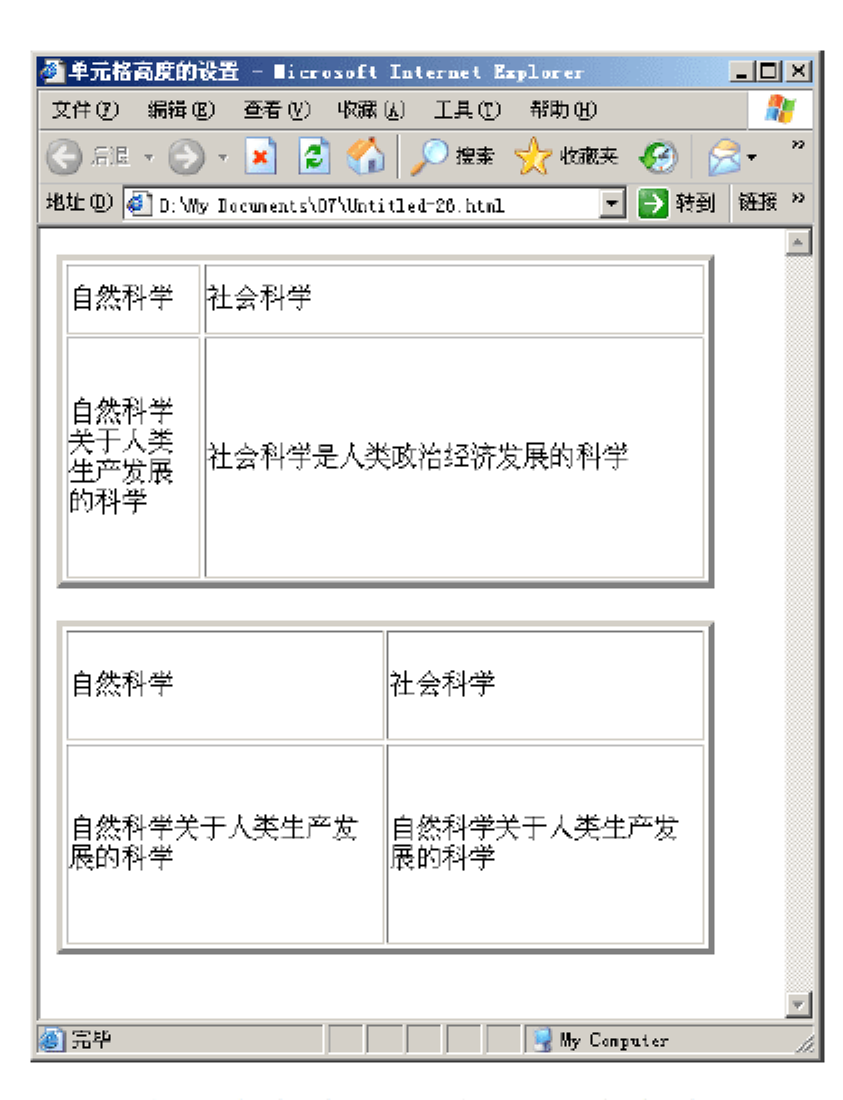


图 7-24 单元格宽度之和大于表格宽度的显示效果

从图 7-23 和图 7-24 可以看出，当单元格的宽度之和大于<table>元素的宽度时，表格的显示宽度为<table>元素中定义的宽度。单元格按照各自的宽度比例自动调整宽度。

7.4.2 高度属性 height

HTML 中用高度属性 height 来定义单元格的高度。语法结构如下所示。

```
<table>
    <tr>
      <td height="数字值">内容部分</td>
    </tr>
</table>
```

其中数字值的实际单位是像素。高度属性与宽度属性基本一样，下面是一个在<td>元素中综合使用 width 和 height 属性的实例。其代码如下所示。

例程 7-24 td-height.html

```
01  <body>
02  <table border=4 bordercolor="#660000" cellspacing=4 cellpadding=9>
03  <caption>某班级的部分成绩</caption>
04  <tr bgcolor="#FFFF00">
              <td width=100 height=30>姓名</td>
              <td width=130>语文</td>
              <td>数学</td>
              <td>英语</td>
              <td width=130>科学</td>
05  </tr>
         <tr>
              <td height=40>张飞</td>
              <td>78</td>
              <td width=130>80</td>
              <td width=70>95</td>
              <td>88</td>
         </tr>
         <tr>
              <td>李明</td>
              <td>80</td>
              <td>85</td>
              <td>91</td>
              <td height=70 width=120>80</td>
         </tr>
         <tr>
              <td>王丽</td>
              <td height=35>87</td>
              <td>76</td>
              <td>91</td>
              <td>89</td>
         </tr>
06  </table>
07  </body>
```

在该实例中，其代码运行后在 IE 浏览器中的显示效果，如图 7-25 所示。



图 7-25　在<td>元素中同时定义 width 和 height 属性后的表现效果

7.4.3　背景颜色属性 bgcolor

为了增加表格的绚丽，可以为不同的单元格分别设置不同的背景颜色。在 HTML 中也可以用背景颜色属性 bgcolor 来定义单元格的背景颜色。语法结构如下所示。

```
<table>
    <tr>
        <td bgcolor="颜色值">内容部分</td>
    </tr>
</table>
```

其中颜色值既可以使用颜色名称，也可以使用 16 进制的颜色值。

下面是一个在<td>元素中使用 bgcolor 属性的实例。其代码如下所示。

例程 7-25　td-bgcolor.html

```
01  <body>
02  <table border=1 bordercolor="#660000" cellspacing=0 cellpadding=5 height=200 width=550>
03  <caption>某班级前两名的成绩</caption>
04  <tr bgcolor="#FFF2C1" >
          <td>姓名</td>
          <td>语文</td>
          <td>数学</td>
          <td>英语</td>
          <td>科学</td>
05  </tr>
          <tr>
          <td bgcolor="#71E2FF">李明</td>
          <td bgcolor="#B382EE">95</td>
          <td bgcolor="#7EF1C0">98</td>
          <td bgcolor="#78DC87">95</td>
          <td bgcolor="#F19B7E">88</td>
          </tr>
          <tr>
```

```
            <td bgcolor="#7EF1CF">张丽</td>
            <td bgcolor="#E49B7E">90</td>
            <td bgcolor="#EF81C6">99</td>
            <td bgcolor="#B6B0E3">91</td>
            <td bgcolor="#B182FF">87</td>
            </tr>
06  </table>
07  </body>
```

在该实例中，04 行对表格的第一行统一设置了颜色，05 行中对各个单元格的颜色都做了设置，其代码运行后的显示效果，如图 7-26 所示。

从图 7-26 可以看出，<td>中定义的背景颜色会更加靠前一些，而且覆盖<tr>元素和<table>元素中定义的背景颜色，希望大家加以注意。



图 7-26　在<td>元素中定义 bgcolor 属性后的表现效果

7.4.4　背景图片属性 background

HTML 中通常用背景图片属性 background 来定义单元格使用的背景图片，在<td>元素中定义的背景图片将覆盖<table>元素中定义的背景颜色或者背景图片，也将覆盖<tr>元素中定义的背景颜色。语法如下所示。

```
<table>
    <tr>
      <td background="图片路径">内容部分</td>
    </tr>
</table>
```

其中图片路径既可以使用相对路径，也可以使用绝对路径。下面是一个在<td>元素中使用 background 属性的实例。其代码如下所示。

例程 7-26　td-background.html

```
01  <body>
02  <table border="4" width="400" height="200" >
03  <tr><td background=" pic1.jpg">蜂鸟</td><td>蜂鸟是世界上最小的鸟类</td></tr>
04  <tr><td>蜂鸟在采集花蜜</td><td background=" pic2.jpg"></td></tr>
05  </table>
06  </body>
```

在该实例中，02 行定义了<table>元素的宽度为 350 像素，高度为 200 像素，边框宽度为 2 个像素，背景颜色为很浅的灰色。03 行和 04 行分别在单元格的左侧和右侧定义了背景图片。其代码运行后的显示效果，如图 7-27 所示。



图 7-27　在<td>元素中定义 background 属性后的表现效果

7.4.5　水平对齐属性 align

在 HTML 中，通常用水平对齐属性 align 来定义单元格内容的水平对齐方式。语法结构如下所示。

```
<table>
  <tr>
    <td>内容部分</td>
  </tr>
</table>
```

align 属性的取值有 left、right 和 center，下面是一个在<td>元素中使用 align 属性的实例。其代码如下所示。

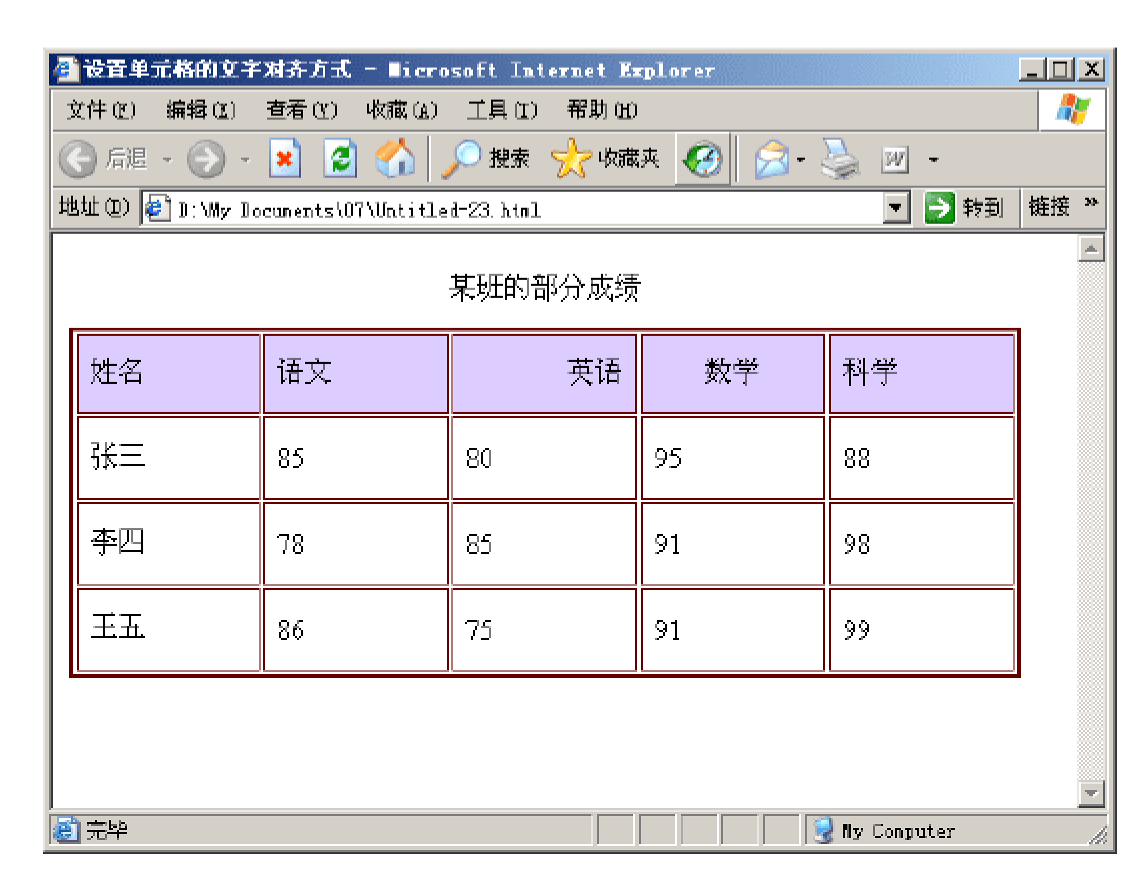
例程 7-27　td-align.html

```
<body>
<table border=2 bordercolor="#660000" cellspacing=2 cellpadding=7 height=200 width=550>
<caption>某班的部分成绩</caption>
<tr bgcolor="#DDCCFF" >
          <td>姓名</td>
          <td align="left">语文</td>
          <td align="right">英语</td>
          <td align="center">数学</td>
          <td>科学</td></tr>
     <tr>
          <td>张三</td>
          <td>85</td>
          <td>80</td>
          <td>95</td>
          <td>88</td></tr>
     <tr>
          <td>李四</td>
```

```
                <td>78</td>
                <td>85</td>
                <td>91</td>
                <td>98</td></tr>
            <tr>
                <td>王五</td>
                <td>86</td>
                <td>75</td>
                <td>91</td>
                <td>99</td></tr>
05  </table>
06  </body>
```

在该实例中，02 行定义了<table>元素的高度为 200 像素，宽度为 550 像素，边框宽度为 2 个像素。04 行在表格的第 1 行第 2 个单元格中，定义了水平对齐属性值为 left。第 3 个单元格中，定义了水平对齐属性值为 right。第 4 个单元格中，定义水平对齐属性为 center。最后一个单元格为默认，其代码运行后的显示效果，如图 7-28 所示。

从图 7-28 可以看出，通过定义<td>元素中的 align 属性，可以更改单元格本身内容的水平对齐方式。



姓名	语文	英语	数学	科学
张三	85	80	95	88
李四	78	85	91	98
王五	86	75	91	99

图 7-28　在<td>元素中定义 align 属性后的表现效果

7.4.6　垂直对齐属性 valign

在 HTML 中，通常用垂直对齐属性 valign 来定义单元格内容的垂直对齐方式。语法结构如下所示。

```
<table>
  <tr>
    <td valign="属性值">内容部分</td>
  </tr>
</table>
```

valign 属性的取值有 top、middle 和 bottom 这 3 种。下面是一个在<td>元素中使用 valign 属性的实例。其代码如下所示。

例程 7-28　td-valign.html

```
<body>
<table border=2 bordercolor="#660000" cellspacing=2 cellpadding=7 height=300 width=500>
<caption>某班的部分成绩</caption>
<tr bgcolor="#00FF00" >
        <td>姓名</td>
            <td valign="top">语文</td>
            <td valign="middle">英语</td>
```

```
                <td valign="bottom">数学</td>
                <td>科学</td>
05    </tr>
          <tr>
                <td>张三</td>
                <td>85</td>
                <td>80</td>
                <td>95</td>
                <td>88</td>
          </tr>
          <tr>
                <td>李四</td>
                <td>78</td>
                <td>85</td>
                <td>91</td>
                <td>98</td>
          </tr>
          <tr>
                <td>王五</td>
                <td>86</td>
                <td>75</td>
                <td>91</td>
                <td>99</td>
          </tr>
06    </table>
07    </body>
```

在该实例中，02 行定义了一个两行两列的<table>元素，其宽度为 500 像素，高度为 300 像素，边框宽度为 2 个像素。04 和 05 行之间定义了表格第 1 行中第 2 个单元格中的垂直对齐属性值为 top。第 3 个单元格中垂直对齐属性值为 middle。第 4 个单元格中垂直对齐属性为 bottom。其代码运行后的显示效果，如图 7-29 所示。

从图 7-29 可以看出，通过定义<td>元素中的 valign 属性，可以更改单元格本身内容的垂直对齐方式。

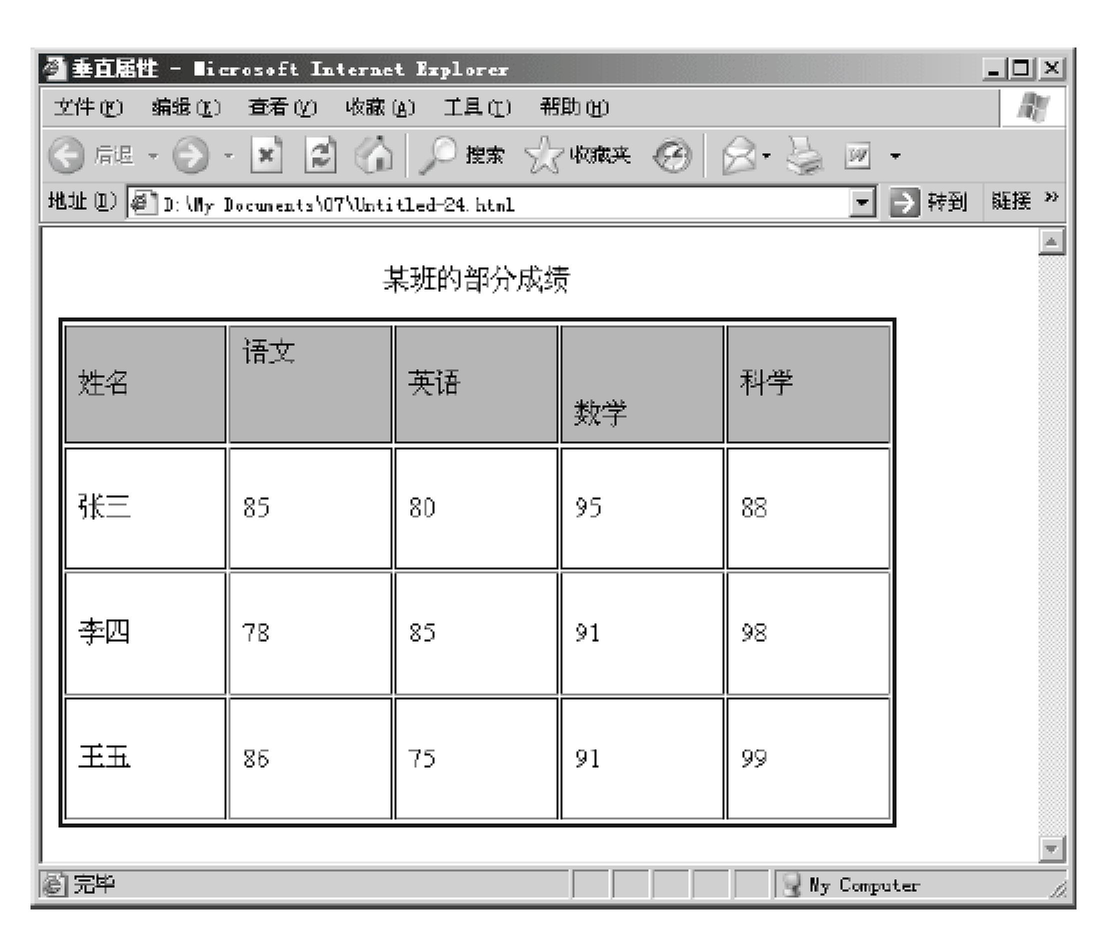


图 7-29　在<td>元素中定义 valign 属性后的表现效果

7.4.7　边框属性 bordercolor

HTML 中也可以用边框属性 bordercolor 来定义单元格的边框颜色。语法结构如下所示。

```
<table>
      <tr>
         <td bordercolor="颜色值">内容部分</td>
      </tr>
</table>
```

其中颜色值既可以使用颜色名称，也可以使用 16 进制的颜色值。下面是一个在<td>元素中使用 bordercolor 属性的实例。其代码如下所示。

例程 7-29　td-bordercolor.html

```
01  <body>
02  <table border="4" width="500" height="200" cellspacing="8">
    <tr>
     <td bordercolor="#FF00FF">自然科学</td><td>社会科学</td>
    </tr>
    <tr>
     <td>物理化学</td><td>政治经济</td>
    </tr>
03  </table>
04  </body>
```

在该实例中，02 行定义了<table>元素的边框宽度为 4 个像素，宽度为 500 像素，高度为 200 像素，单元格间距为 8。第一行第一个单元格中定义边框颜色为紫色色。其代码运行后的显示效果，如图 7-30 所示。

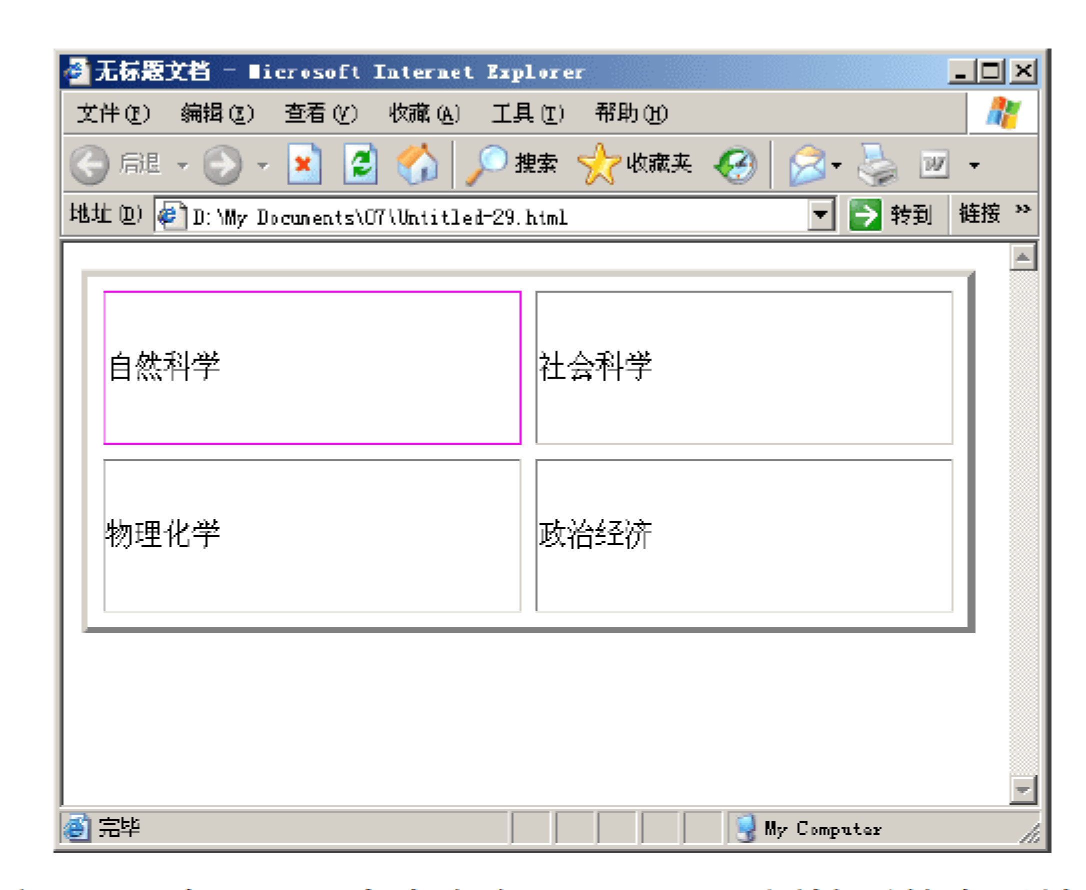


图 7-30　在<td>元素中定义 bordercolor 属性后的表现效果

7.4.8　合并列属性 colspan

合并列属性 colspan 用来将某一行的几个单元格合并成一个。语法结构如下所示。

```
<table>
    <tr>
        <td colspan="数字值">内容部分</td>
    </tr>
</table>
```

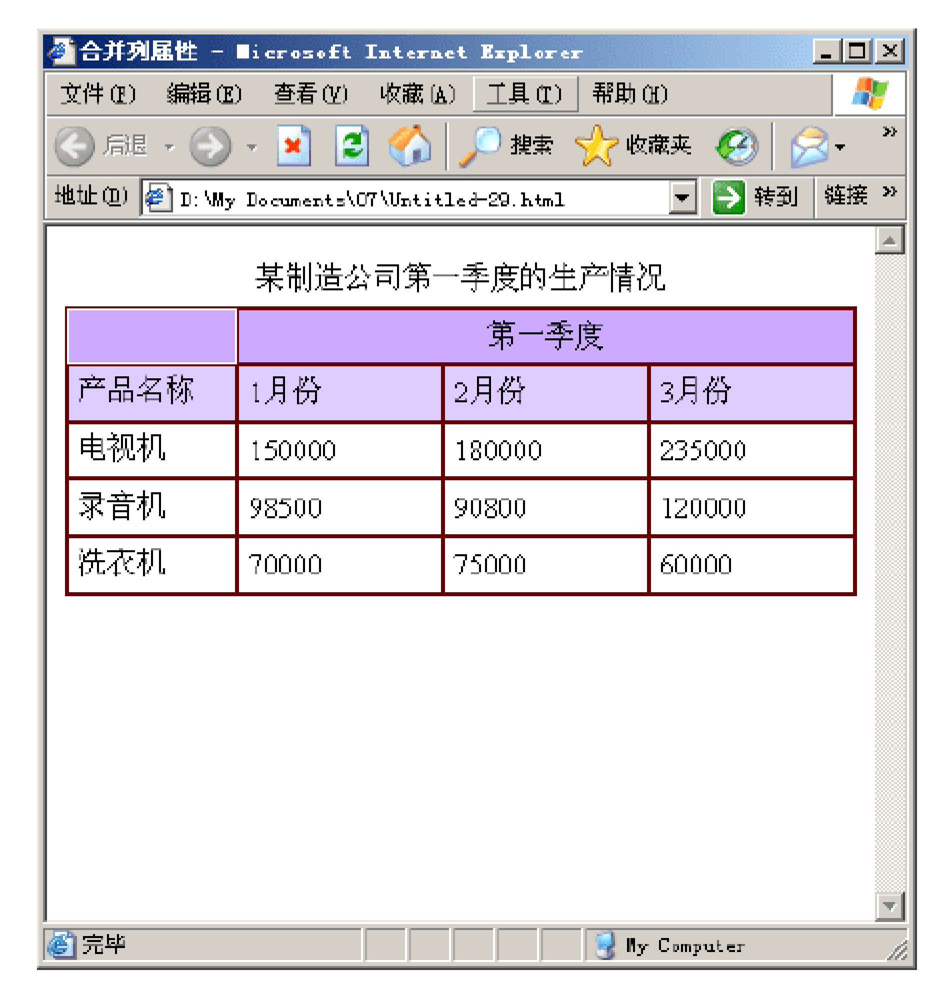
其中数字值的含义是合并的列的数目。下面是一个在<td>元素中使用 colspan 属性的实例。其代码如下所示。

例程 7-30　td-colspan.html

```
<body>
<table border=1 bordercolor="#660000" cellspacing=0 cellpadding=5>
<caption>某制造公司第一季度的生产情况</caption>
<tr bgcolor="#CCAAFF">
            <td width=80></td>
            <td colspan=3 width=400 align="center">第一季度</td>
</tr>
<tr bgcolor="#DDCCFF">
            <td>产品名称</td>
```

```
                <td>1 月份</td>
                <td>2 月份</td>
                <td>3 月份</td>
07      </tr>
            <tr>
                <td>电视机</td>
                <td>150000</td>
                <td>180000</td>
                <td>235000</td>
            </tr>
            <tr>
                <td>录音机</td>
                <td>98500</td>
                <td>90800</td>
                <td>120000</td>
            </tr>
            <tr>
                <td>洗衣机</td>
                <td>70000</td>
                <td>75000</td>
                <td>60000</td>
            </tr>
08      </table>
09      </body>
```

该示例定义了 04 行中定义了第 1 行的第 2 个单元格将 3 列合并，其代码运行后的显示效果，如图 7-31 所示，其中第 1 行的第 2 个单元格跨了 3 列。



某制造公司第一季度的生产情况

	第一季度		
产品名称	1月份	2月份	3月份
电视机	150000	180000	235000
录音机	98500	90800	120000
洗衣机	70000	75000	60000

图 7-31　在<td>元素中定义 colspan 属性后的表现效果

7.4.9　合并行属性 rowspan

单元格除了可以在水平方向下跨列，还可在垂直方向上跨行。HTML 中通常用合并行属性 rowspan 来将某一列的几个单元格合并成一个。注意 rowspan 设置的是单元格在垂直方向上跨行的个数。也可以说是单元格向下打通的单元格个数。语法结构如下所示。

```
<table>
    <tr>
      <td rowspan="数字值">内容部分</td>
    </tr>
</table>
```

其中数字值的含义是合并的行的数目。下面是一个在<td>元素中使用 rowspan 属性的实例。其代码如下所示。

例程 7-31　td-rowspan.html

```
01  <body>
02  <table border=1 bordercolor="#660000" cellspacing=0 cellpadding=5>
03  <caption>某图书馆的销售分类</caption>
04  <tr bgcolor="#FF00FF">
            <td width=130>类别</td>
            <td width=290>子类别</td>
05  </tr>
06  <tr><td rowspan=2>漫画区</td><td>最新漫画展示区</td></tr>
07  <tr><td>漫画制作教程</td></tr>
08  <tr><td rowspan=4>考试专区</td><td>小学升初中专区</td></tr>
09  <tr><td>中高考专区</td></tr>
10  <tr><td>考研专区</td></tr>
11  <tr><td>公务员专区</td></tr>
12  </table>
13  </body>
```

在该实例中，02 行定义了一个两行两列的<table>元素，边框宽度为 2 个像素。06 行在表格第 2 行第 1 个单元格中定义了 rowspan 属性值为 2，08 行在表格第 3 行第 1 个单元格中定义了 rowspan 属性值为 4。其代码运行后的显示效果，如图 7-32 所示。

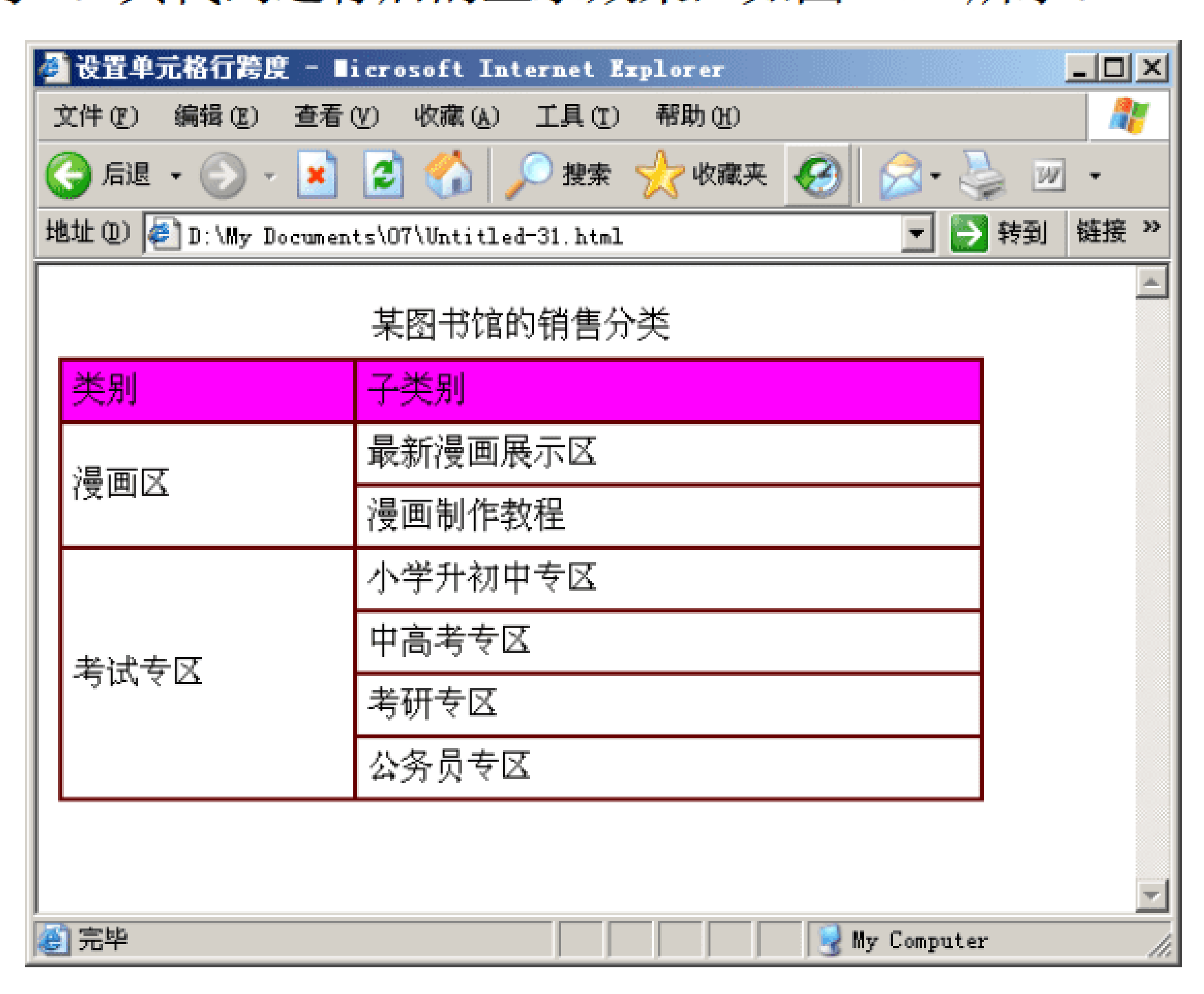


图 7-32　在<td>元素中定义 rowspan 属性后的表现效果

从图 7-32 可以看出第 2 行的第 1 个单元格合并了 2 行，第 3 行的第 1 个单元格合并了 4 行。

7.4.10　同行显示属性 nowrap

当表格没有设定宽度的时候，整个表的宽度会根据表格内容进行调整，但表格的宽度不会超出浏览器的宽度。当单元格内的内容过长的时候会自动换行，如果不希望表格的某个单元格的内容换行，可以通过同行显示属性 nowrap 来使单元格中的文本同行显示。只有当文本中出现换行元素
，或者其他默认换行的元素时文本内容才能换行显示。语法结构如下所示。

```
<table>
    <tr>
        <td nowrap=" nowrap">内容部分</td>
    </tr>
</table>
```

先看下面的这个实例，其代码如下所示。

例程 7-32 td-nowrap.html

```
<body>
<table align="center" border=5 cellspacing=0 bordercolor="#990000" cellpadding=3>
<caption>几本书的简介</caption>
<tr><th>书名</th><th>作者</th><th>内容简介</th></tr>
<tr>
            <td>史记</td>
            <td>司马迁</td>
            <td>《史记》是中国西汉时期的历史学家司马迁撰写的史学名著，列“二十四史"之首，
《史记》是中国古代最著名的古典典籍之一。
    </td>
</tr>
<tr>
            <td>水浒</td>
            <td>施耐庵</td>
            <td>元末明初小说家施耐庵根据民间流传的宋江起义故事，写成长篇小说《水浒传》。</td>
</tr>
<tr>
            <td>红楼梦</td>
            <td>曹雪芹</td>
            <td>书中以贾、史、王、薛四大家族为背景，以贾宝玉、林黛玉爱情悲剧为主线，着重描
写贾、宁二府由盛到衰的过程。</td>
</tr>
</table>
</body>
```

在该实例中，其代码运行后的显示效果，如图 7-33 所示。因为没有使用 nowrap 属性，单元格中的文本内容根据页面大小自动换行。

如果不希望表格的某个单元格的内容换行，可以通过 nowrap 参数进行设置，实例代码如下。

例程 7-33 table-nowraptwo.html

```
<body>
<table align="center" border=5 cellspacing=0 bordercolor="#990000" cellpadding=3>
<caption>几本书的简介</caption>
        <tr>
            <th>书名</th>
            <th>作者</th>
            <th>内容简介</th>
        </tr>
<tr>
            <td>史记</td>
            <td>司马迁</td><td nowrap>《史记》是中国西汉时期的历史学家司马迁撰写的史学名著，
列“二十四史"之首，《史记》是中国古代最著名的古典典籍之一。</td>
</tr>
```

```
<tr><td nowrap >水浒传</td>
        <td>施耐庵</td>
        <td>元末明初小说家施耐庵根据民间流传的宋江起义故事，写成长篇小说《水浒传》。</td>
</tr>
<tr><td nowrap >红楼梦</td>
        <td>曹雪芹</td>
        <td>书中以贾、史、王、薛四大家族为背景，以贾宝玉、林黛玉爱情悲剧为主线，着重描写贾、宁二府由盛到衰的过程。</td>
</tr>
</table>
</body>
```

在该实例中，代码运行后的显示效果如图 7-34 所示。

从图 7-33 和图 7-34 可以看出，运行代码看到表格中设置了 nowrap 参数的单元格内容不换行显示了，而超出浏览器宽度的内容则通过浏览器的滚动条来显示。

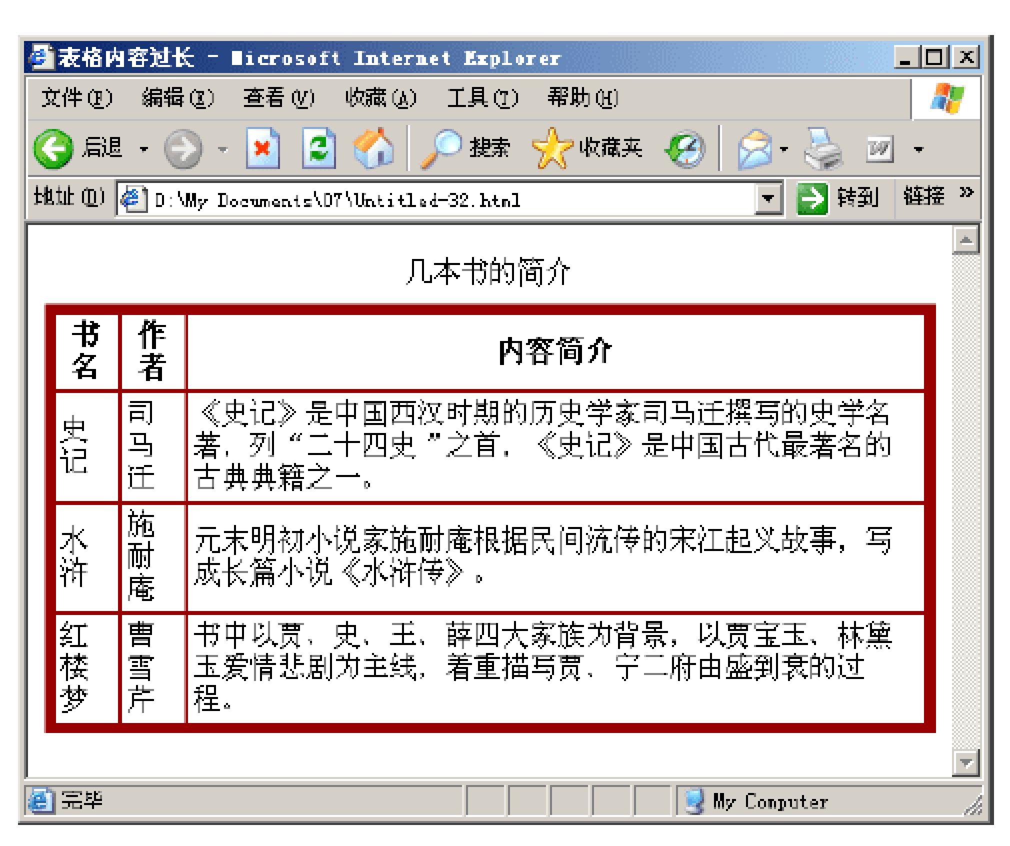


图 7-33　在<td>元素中没有定义 nowrap 属性后的表现效果图

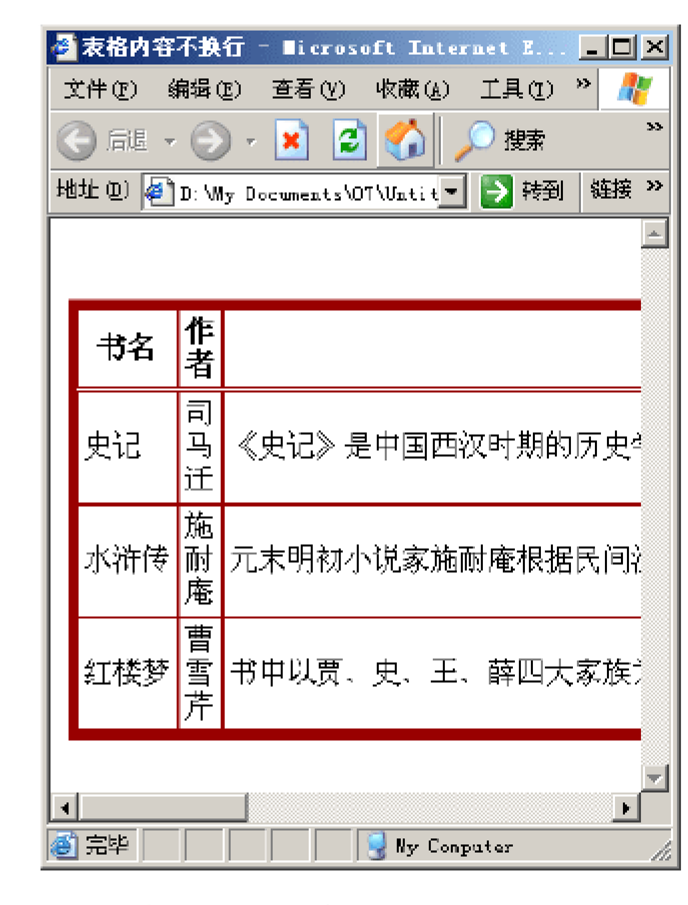


图 7-34　在<td>元素中定义 nowrap 属性

7.5　表格中使用的其他元素

表格中除了使用<tr>元素和<td>元素以外，还可以使用其他的元素，其中包括<caption>、<th>、<thead>等。下面将逐一详细讲解。

7.5.1　表格标题元素<caption>

在制作表格时，可以用表格标题元素<caption>来定义表格的标题。<caption>元素中定义的表格标题内容，将显示在表格的顶部(表格边框的外部)。语法结构如下所示。

```
<table>
  <caption>表格标题</caption>
    <tr>
      <td>内容部分</td>
```

```
        </tr>
    </table>
```

<caption>元素必须添加在表格元素<table>和行元素<tr>之间，或者两个行元素<tr>之间。<caption>元素的放置位置并不影响其显示效果(将在实例 7-36 中演示)。

下面是一个在表格中使用<caption>元素的实例。其代码如下所示。

例程 7-34　caption.html

```
01  <body>
02  <table border=1 bordercolor="#660000" cellspacing=0 cellpadding=5>
03  <caption>某图书馆的销售分类</caption>
04  <tr bgcolor="#FF00FF">
            <td >类别</td>
            <td >子类别</td>
        </tr>
        <tr>
            <td >漫画区</td>
            <td>最新漫画展示区</td>
        </tr>
        <tr>
            <td >考试专区</td>
            <td>小学升初中专区</td>
        </tr>
05  </table>
06  </body>
```

在该实例中，03 行将<caption>元素放置在表格元素<table>和行元素<tr>之间。其代码运行后的显示效果，如图 7-35 所示。

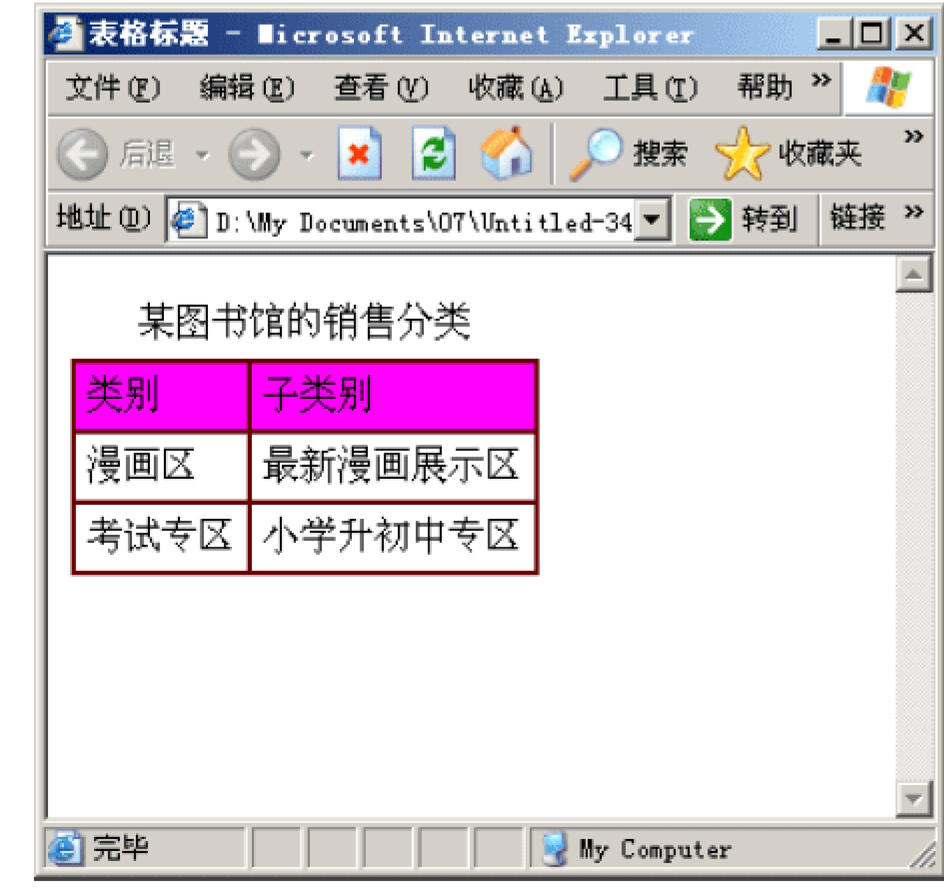


图 7-35　在表格中使用<caption>元素的表现效果

7.5.2　表格头部元素<th>

表格头部元素<th>用来定义表格的表头。<th>元素和<td>元素类似，只是使用<th>元素的文本内容会加粗显示，同时默认为中间对齐。语法结构如下所示。

```
<table>
  <tr>
    <th>内容部分</th>
  </tr>
</table>
```

下面是一个在表格中使用<th>元素的实例。其代码如下所示。

例程 7-35　th.html

```
<body>
<table border="4" width="400" height="200">
<tr><th>自然科学</th><th>社会科学</th></tr>
<caption>th 和 td</caption>
```

```
<tr><td>自然科学</td><td>社会科学</td></tr>
</table>
</body>
```

其代码运行后的显示效果，如图 7-36 所示。



图 7-36　在表格中使用<th>元素的表现效果

<th>元素中可以使用的所有属性如表 7-5 所示。

表 7-5　<th>元素的所有属性

属 性 名 称	写　法
文本显示方向属性	dir
指定语言属性	lang
类属性	class
定义级联样式属性	style
对齐属性	align
标题属性	title
取消自动换行属性	nowrap
标记属性	id
背景颜色属性	bgcolor
背景图片属性	background
垂直对齐属性	valign
边框颜色属性	bordercolor
暗边框属性	bordercolordack
亮边框属性	bordercolorright
合并列属性	colspan
合并行属性	rowspan
表头名称属性	headers

7.5.3 表格头行元素<thead>

表格头行元素<thead>用来定义表格最上端表首的样式。其中可以设置背景颜色、文字对齐方式、文字的垂直对齐方式等。语法结构如下所示。

```
<table>
 <thead bgcolor="颜色代码" align="对齐方式" valign="垂直对齐方式">
<tr>
    <th>内容部分</th>
      </tr>
  </thead>
</table>
```

在该语法中 bgcolor、align、valign 参数的取值范围与单元格中的设置方法相同，align 可以取 left、center 或 right；valign 可以取 top、middle 或 bottom。注意在<thead>标记内还可以包含<td>、<th>和<tr>标记。而一个表元素中只能有一个<thead>标记。下面是一个在表格中使用<thead>元素的实例。其代码如下所示。

例程 7-36 thead.html

```
01  <body>
02  <table align="center" border=1 bordercolor="#990000" cellpadding=3 width=500 height=180>
03  <caption>春节预备购买物品</caption>
04  <thead bgcolor="#00FFFF" align="center" valign="bottom">
05  <tr><th>物品名</th><th>类型</th><th>用途</th><th>价格预算</th><th>数量</th></tr>
06  </thead>
07  <tr><td>金龙鱼油</td><td>食品</td><td>生活必须品</td><td>150</td><td>3</td></tr>
08  <tr><td>礼花炮</td><td>娱乐品</td><td>增加节日气氛</td><td>200</td><td>15 个</td></tr>
09  <tr><td>红包</td><td>礼品</td><td>送人</td><td>200</td><td>25</td></tr>
10  </table>
11  </body>
```

04 行中定义了表格最上端的颜色为蓝绿色，居中。其代码运行后的显示效果，如图 7-37 所示。



图 7-37 在表格中使用<thead>元素的表现效果

7.5.4 表主体元素<tbody>

表主体元素<tbody>用来将表格的内容分隔成各个独立的部分。可以在每个独立的部分中定义特定的表现效果。语法结构如下所示。

```
<table>
<tbody bgcolor="颜色代码" align="对齐方式" valign="垂直对齐方式">
    <tr>
      <th>内容部分</th>
    </tr>
</tbody>
</table>
```

<tbody>元素的用法与<thead>类似，下面是一个在表格中使用<tbody>元素的实例。其代码如下所示。

例程 7-37 tbody.html

```
01 <body>
02 <table align="center" border=4 bordercolor="#990000" cellpadding=3 width=500 height=180>
03 <caption>某学校图书馆借书记录</caption>
04 <thead bgcolor="#97B6FF" align="center" valign="bottom">
05 <tr><th>书籍名</th><th>借书日期</th><th>借书人姓名</th><th>借书人专业</th><th>数量</th></tr>
06 </thead>
07 <tbody bgcolor="#FFFF00" align="left" valign="bottom">
08 <tr><td>网站管理与设计</td><td>3 月 24 日</td><td>张三</td><td>计算机信息技术
</td><td>1</td></tr>
09 <tr><td>工程力学</td><td>3 月 24 日</td><td>李明</td><td>力学系</td><td>1</td></tr>
10 <tr><td>理论物理</td><td>3 月 24 日</td><td>李静</td><td>物理系</td><td>1</td></tr>
11 </tbody>
12 </table>
13 </body>
```

07 行和 11 行中对表格做了统一的设置，运行这段代码，可以看到表格的主体内容统一设置了样式，如图 7-38 所示。



图 7-38 在表格中使用<tbody>元素的表现效果

<tbody>元素中可以使用的所有属性如表 7-6 所示。

表 7-6 <tbody>元素的所有属性

属 性 名 称	写　法
文本显示方向属性	dir
指定语言属性	lang
类属性	class
定义级联样式属性	style
对齐属性	align
标题属性	title
垂直对齐属性	valign

7.5.5 表格行尾元素<tfoot>

表格行尾元素<tfoot>用来定义表格行尾的标注等内容。语法结构如下所示。

```
<table>
  <tfoot bgcolor="颜色代码" align="对齐方式" valign="垂直对齐方式">
    <tr>
      <td>内容部分</td>
    </tr>
  </tfoot>
</table>
```

在该语法中 bgcolor、align、valign 参数的取值范围与<thead>标记中的设置方法相同。一个表元素中只能有一个<tfoot>标记。下面是一个在表格中使用<tfoot>元素的实例。其代码如下所示。

例程 7-38 tfoot.html

```
<body>
<table align="center" border=4 bordercolor="#990000" cellpadding=3 width=450 height=160>
<caption>某学校图书馆借书记录</caption>
<thead bgcolor="#97B6FF" align="center" valign="bottom">
<tr><th>书籍名</th><th>借书日期</th><th>借书人姓名</th><th>借书人专业</th><th>数量</th></tr>
</thead>
<tbody bgcolor="#FFFF00" align="left" valign="bottom">
<tr><td>网站管理与设计</td><td>3 月 24 日</td><td>张三</td><td>计算机信息技术</td><td>1</td></tr>
<tr><td>工程力学</td><td>3 月 24 日</td><td>李明</td><td>力学系</td><td>1</td></tr>
<tr><td>理论物理</td><td>3 月 24 日</td><td>李静</td><td>物理系</td><td>1</td></tr>
</tbody>
<tfoot bgcolor="#FAA9AB" align="right" valign="middle">
<tr><td colspan=5>表格创建日期：XX-XX-XX</td></tr>
</tfoot>
</table>
</body>
```

其代码运行后的显示效果，如图 7-39 所示。



图 7-39　在表格中使用<tfoot>元素的表现效果

<tfoot>元素中可以使用的所有属性如表 7-7 所示。

表 7-7　<tfoot>元素的所有属性

属性名称	写　法
文本显示方向属性	dir
指定语言属性	lang
类属性	class
定义级联样式属性	style
对齐属性	align
标题属性	title
垂直对齐属性	valign

7.6 本章习题

一、选择题

1．关于表格的描述正确的一项是(　　)。

A. 在单元格内不能继续插入整个表格

B. 可以同时选定不相邻的单元格

C. 粘贴表格时，不粘贴表格的内容

D. 在网页中，水平方向可以并排多个独立的表格

2．如果一个表格包括有 1 行 4 列，表格的总宽度为 699 像素，间距为 5 像素，填充为 0 像素，边框为 3 像素，每列的宽度相同，那么应将单元格定制为(　　)像素宽。

A. 126　　B. 136　　C. 147　　D. 167

3．要使表格的边框不显示，应设置 border 的值是(　　)。

A. 1　B. 0　C. 2　D. 3

4．用于设置表格背景颜色的属性的是（　　)。

A. background　B. bgcolor　C. BorderColor　D. backgroundColor

5．以下标记中，用于定义一个单元格的是(　　)。

A. <td> </td>　　B. <tr>…</tr>

C. <table>…</table>　　D. <caption>…</caption>

6．以下说法中，错误的是(　　)。

A. 表格在页面中的对齐应在 TABLE 标记符中使用 align 属性

B. 要控制表格内容的水平对齐，应在 TR、TD、TH 中使用 align 属性

C. 要控制表格内容的垂直对齐，应在 TR、TD、TH 中使用 valign 属性

D. 表格内容的默认水平对齐方式为居中对齐

二、填空题

1．表格的标签是______，单元格的标签是______。

2．表格的宽度可以用百分比和_____两种单位来设置。

3．表格有 3 个基本组成部分：行、列和____。

4．表格中用列组标记符是_________。

5．将表格的行分组，用到的主要标记是___________。

6．要控制单元格内容与表格框线之间的空白，应在 TABLE 标记符中使用属性___________。

7．要使整个表格行采用某种对齐方式，应在_____________标记符中使用 aligh 属性。

三、实战练习

1．制作一个 3 行 4 列的表格。

2．制作一个 4 行 5 列的表格，要求使用水平或者竖直方向上的合并。

3．制作一个表格，并为各个单元格设置颜色，并加上制作日期。

4．编写出实现如图 7-40 所示页面效果的关键 html 代码。其中，A、B、C、D、E 均为默认字号和默认字体，并且加粗显示，它们都位于各自单元格的正中间，A 单元格的高度为 200 像素，B 单元格的高度为 100 像素，C 单元格的宽度为 100 像素，高度为 200 像素。

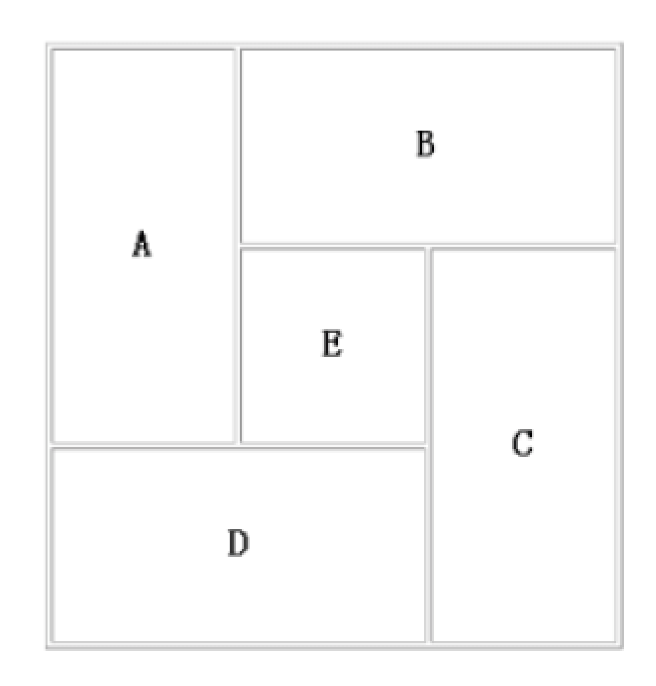


图 7-40　页面效果

5．已知页面效果如图 7-41 所示(其中的细线效果均为 1 像素粗细，颜色为黑色)，请填写以下 html 代码中留下的空白。

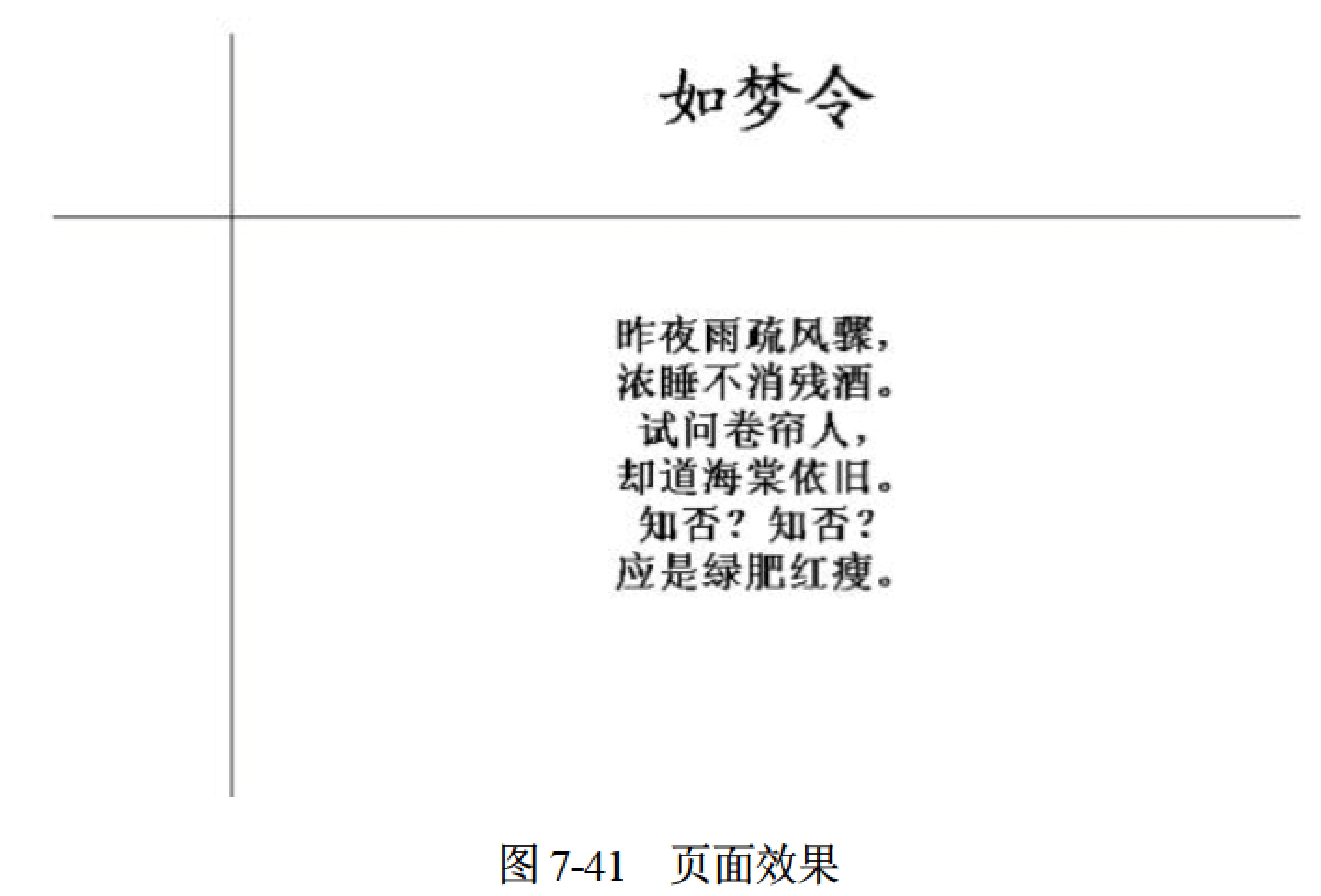


图 7-41　页面效果

```
<TABLE cellspacing=____ cellpadding=______ align="center">
<TR height="100">
  <TD width="100"> 
  <TD width="1"_____________>
  <TD width="600"><H1 align="center"><FONT _______=" 楷 体 _gb2312"> 如 梦 令 </FONT></H1>
<TR ________="1">
  <TD colspan="3" ____________>
<TR height="600">
  <TD width="100"> 
  <TD width="1" ____________>
  <TD width="600" align="middle" valign="top"><H3><BR><BR>昨夜雨疏风骤，<BR>浓睡不消残酒。<BR>试问卷帘人，<BR>却道海棠依旧。<BR>知否？知否？<BR>应是绿肥红瘦。</H3>
```

</TABLE>

(1)________________

(2)________________

(3)________________

(4)________________

(5)________________

Chapter 8

链接元素

平时在浏览网页时，当单击某个字或某个图片时，就可以打开另外一个画面。这个作用对网页来说相当重要，否则游览者只能每打开一个页面时都要在地址栏内输一次地址。

本章主要内容有:

◎ 链接元素各种属性的使用方法和表现效果。

◎ 理解两种路径的特点和用法。

◎ 能够使用图片进行链接。

8.1 链接和路径

本节主要讲解关于超级链接的概念、路径基础知识、相对路径和绝对路径等内容。下面分别进行介绍。

8.1.1 超链接的概念

超链接简单地说就是从一个网页转到另一个网页的途径。互联网就是通过各种超链接把整个网站的信息有机地结合到一起。使浏览者从一个页面跳转到另一个页面，实现文档互联、网站互联。

超文本链接(hypertextlink)通常简称为超链接(hyperlink)，或者简称为链接(link)。链接是HTML 的一个最强大和最有价值的功能。链接是指文档中的文本或者图像与另一个文档、文档的一部分或者一幅图像链接在一起。其基本语法如下。

```
<a href="文件名">链接元素</a>或<a href="URL">链接元素</a>
```

在该语法中，链接元素可以是文字，也可以是图片或其他页面元素。其中 href 是 hypertext reference 的缩写。通过超级链接的方式可以使各个网页之间连接起来，使网站中众多的页面构成一个有机整体，使访问者能够在各个页面之间跳转。超级链接可以是一段文本、一幅图像或其他网页元素，当在浏览器中用鼠标单击这些对象时，浏览器可以根据指示载入一个新的页面或者转到页面的其他位置。

下面是一个使用超链接的实例，其代码如下所示。

例程 8-1 example.html

```
<body>
点击 <a href="http://www.hao123.com" >hao123 首页</a> 这个超链接，可以打开 hao123 的首页。
</body>
```

该实例中，定义了一个超链接用来链接 hao123 站点的首页。其运行后的显示效果如图 8-1 所示。链接内容以有颜色的字显示，用来区分其他没有链接的内容。当用鼠标单击含有链接的内容后，就会跳转到 hao123 的首页，其显示效果如图 8-2 所示。

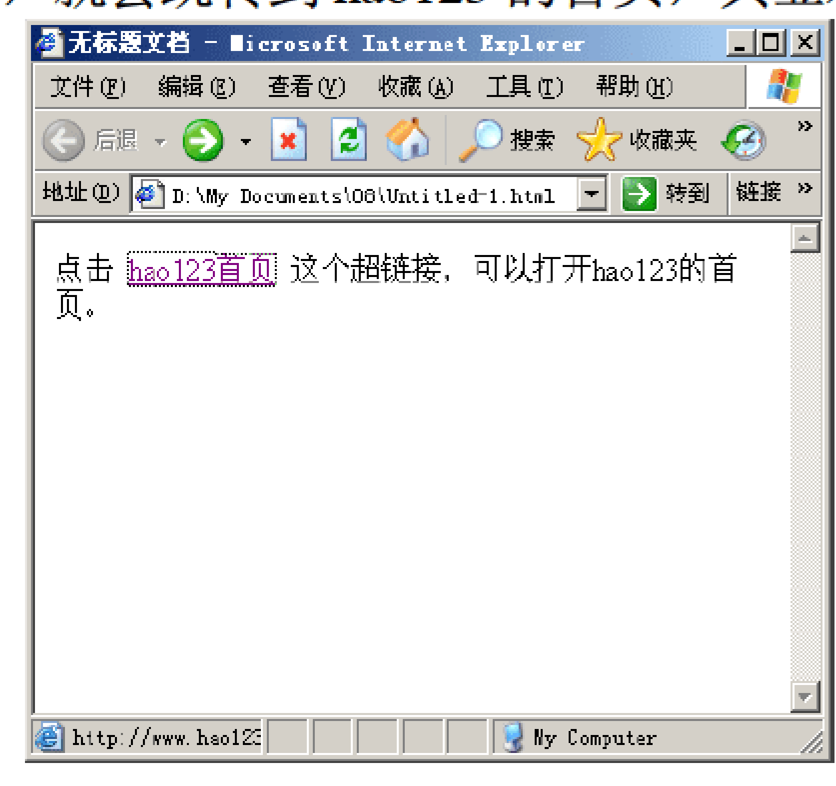


图 8-1 使用超链接的显示效果



图 8-2 使用超链接打开新页面的显示效果

8.1.2 路径 url

路径 url 用来定义一个文件、内容或者媒体等的所在地址。这个地址可以是相对链接，也可以是一个网络中的绝对地址。下面是一个使用路径的实例。其代码如下所示。

例程 8-2 url.html

```
01  <body>
02  图片中的路径：<img src="pic.jpg" width="150" />
03  <br /><br />
04  背景中的路径：<table border="2" width="240" height="100"
    background="http://www.hao123.com/images/logo_140.gif">
05  <tr>
06  <td>链接的三种情况</td>
07  </tr>
08  </table>
09  <br />
10  链接中的路径：<a href="http://www.hao123.com">链接内容</a>
11  </body>
```

该实例中，02、04、10 行是在各种页面元素中使用路径的方法。其运行后的显示效果如图 8-3 所示。

再来看两行代码，这两行代码里应用了绝对链接的写法。

```
<img src="F:\HTml\html 跟我学\pic.jpg" width="150" />
<a href="http://www.hao123.com">
```

绝对链接也称为绝对路径，绝对路径就是主页上的文件或目录在硬盘上真正的路径。使用绝对路径定位链接目标文件比较清晰，但是绝对路径有两个缺点：一是需要输入更多的内容，二是如果该文件被移动了，就需要重新设置所有的相关链接。当然使用网上地址是没问题的，但是上两行代码中的图片到了网上就不可用了。这是因为路径设置的问题。这里的路径为“F:\HTml\html 跟我学\pic.jpg”，在本地确实可以找到，但是到了网站上该文件不一定在这个路径下，所以就会出问题。这时候就需要用到相对路径，相对路径用来指定内容的相对地址。这个地址可以是相对于文档所在的当前目录，也可以是相对于指定的基本的路径。在相对路径中，一般可以忽略 HTTP 协议和域名等内容，直接使用文件的相对地址就可以。例如，可以使用如下所示的链接路径。



图 8-3 各种元素中使用路径的显示效果

```
<a href="images/pic.jpg">链接内容</a>
```

下面是一个使用相对路径的实例。其代码如下所示。

例程 8-3 relative-url.html

```
<body>
链接中使用相对路径：<a href="pic.jpg" target="_blank">……</a>
</body>
```

在该实例中，链接的地址是一个相同文件夹里的图片文件。其代码运行后，单击链接文本时的显示效果如图 8-4 所示。

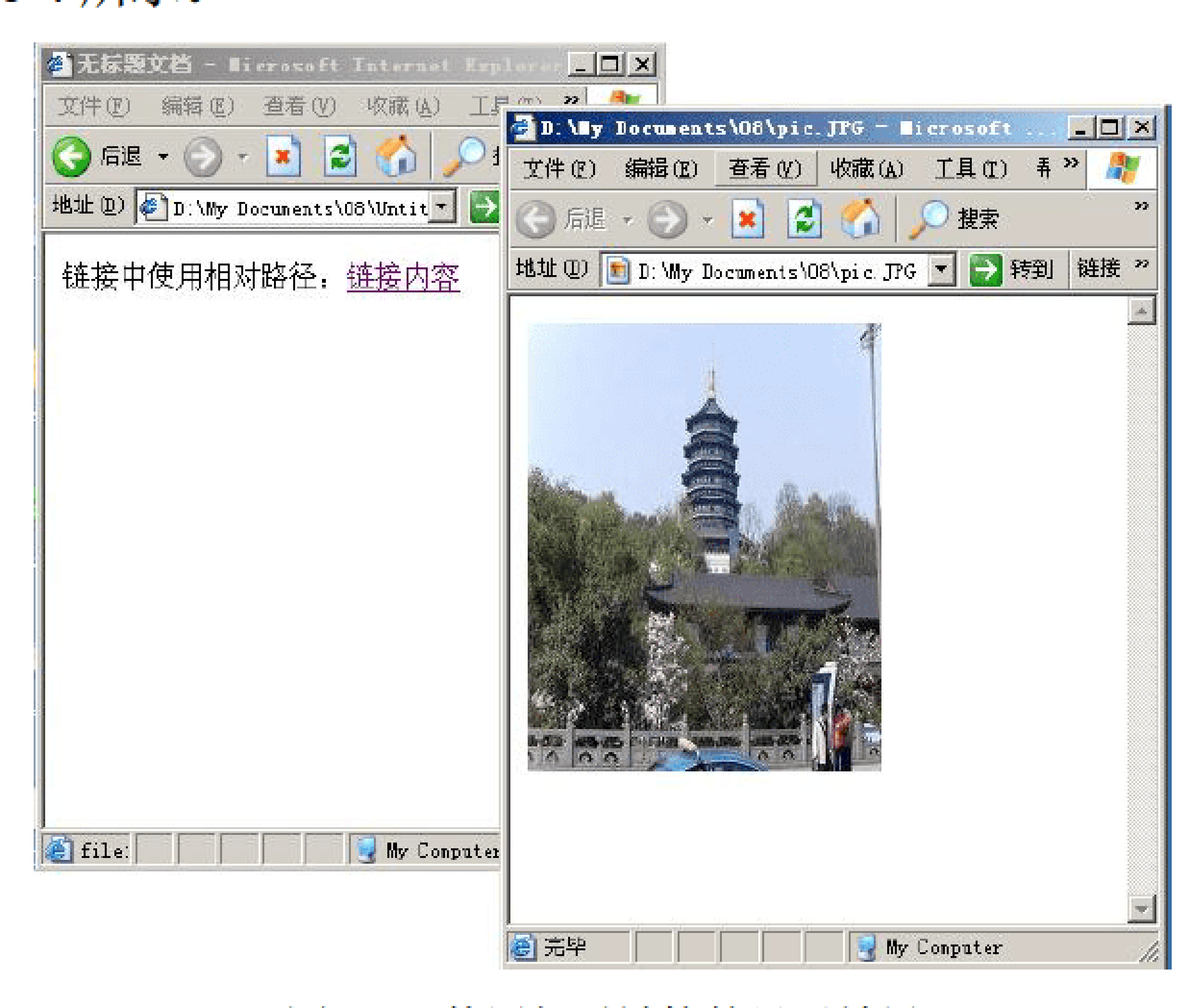


图 8-4 使用相对链接的显示效果

8.1.3 HTTP 路径

HTTP 路径用来链接 Web 服务器中的文档。下面是 HTTP 路径可以使用的写法。

```
http://域名/目录/目标文件
http://域名/目录/目标文件#片段
http://域名/目录/目标文件？变量名=变量参数
```

其中第一种链接是最普通的链接格式，不使用任何参数和变量。第二种是链接到目标文件的某个一个片段的位置上。第三种是传递某个参数，使用相应的程序来处理相关内容。下面是一个使用普通 HTTP 路径的实例。其代码如下所示。

例程 8-4 http.html

```
<body>
链接地址：<a href="http://www.baidu.com/index.php?tn=myppcb&bar=11">链接</a>
</body>
```

其代码运行后，当单击含有链接的文本时，显示效果如图 8-5 所示。

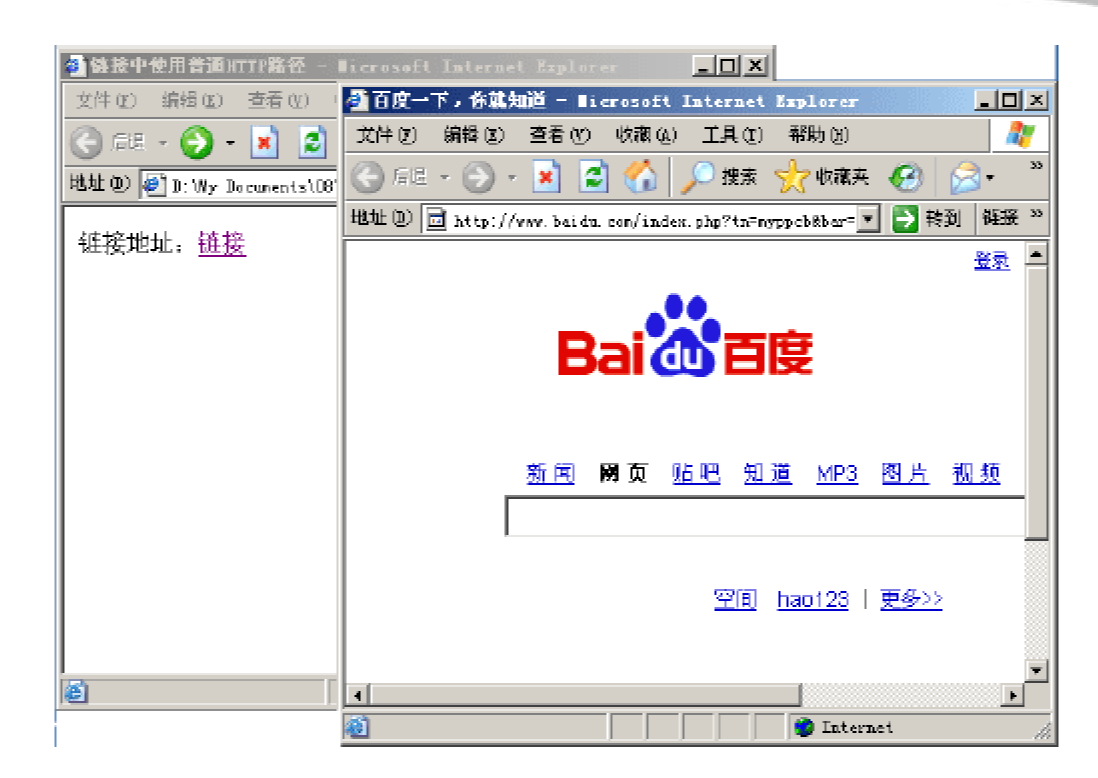


图 8-5 使用普通 HTTP 路径的显示效果

8.1.4 FTP 路径

FTP 路径用来链接 FTP(File Transfer Protocol，文件传输协议)服务器中的文档。下面是 FTP 路径可以使用的写法。

```
<A href="ftp://……">链接文字</A>
```

其中域名和 IP 地址其实是服务器地址的两种写法，都代表网络中唯一的标识。如果要在网页中使用 FTP 链接，要使用如下的写法。

```
ftp://ftp 域名/目录
```

使用 FTP 路径时，可以在浏览器中直接输入相应的 FTP 地址，打开相应的目录或者下载相关内容。也可以使用相关的软件，打开 FTP 地址中相应的目录或者下载相关内容。

下面是一个使用 FTP 路径的实例。其代码如下所示。

例程 8-5 ftp.html

```
<body>
链接中使用邮件路径：<a href="ftp://ftp.zju.edu.cn" target="_blank">链接内容</a>
</body>
```

该实例中，链接了浙江大学的 FTP 站点。其代码运行后，当单击含有链接的文本时，显示效果如图 8-6 所示。

从图 8-6 可以看出，此时打开一个新的窗口，显示了此目录下所有的文件，可以使用复制、粘贴功能，或者借助下载软件来下载目录中的内容。

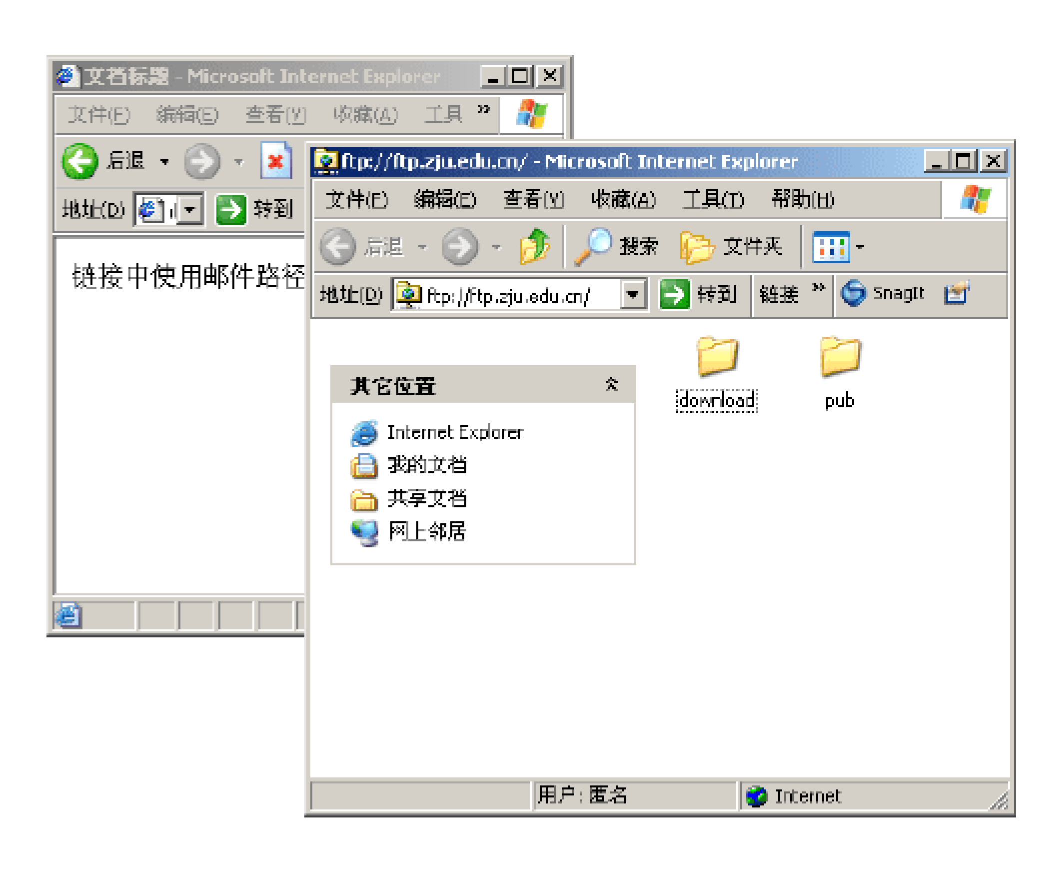


图 8-6 使用 FTP 路径路径的显示效果

8.1.5 邮件路径

邮件路径用来链接一个电子邮件的地址。下

面是邮件路径可以使用的写法。

```
mailto:邮件地址
```

下面是一个使用邮件路径的实例。其代码如下所示。

例程 8-6　email.html

```
<body>
使用邮件路径：<a href="mailto:sucess1123@126.com">链接</a>
</body>
```

其代码运行后，当单击含有链接的文本时，显示效果如图 8-7 所示。

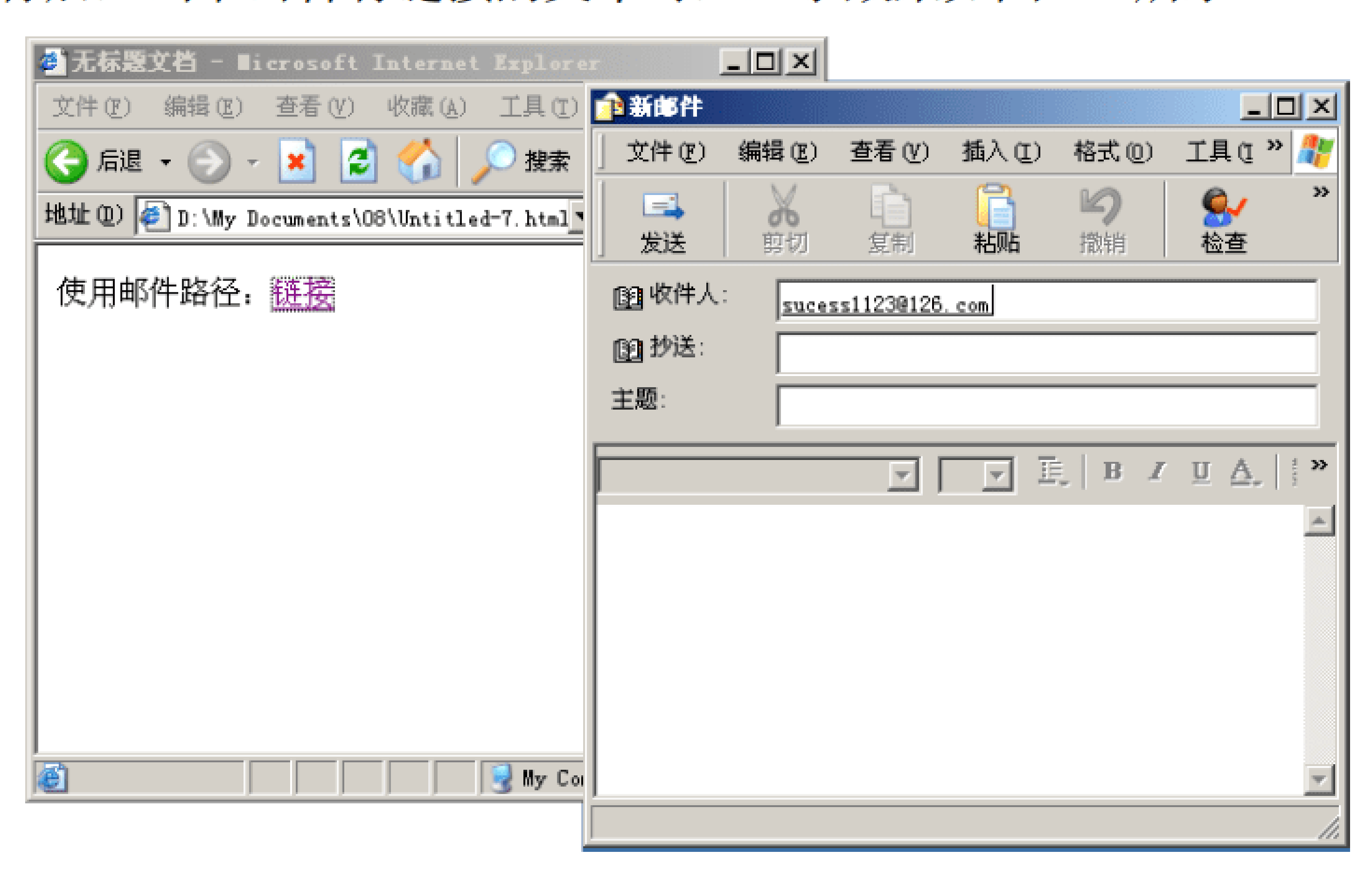


图 8-7　使用邮件路径的显示效果

当单击含有链接的内容后，会弹出一个发送邮件的对话框(实例中使用的是 Windows 操作系统)。

8.2 链接元素<a>

链接元素<a>用来定义一个超链接。8.1 节使用链接时已经用到<a>元素，其实在<a>元素中，不但可以包含文本内容，也可以包含图片等其他内容。语法结构如下所示。

```
<a href="链接的路径">链接的文本</a>
```

在<a>元素中，一般要指定 href 属性，用来指向链接的目标。下面是一个使用<a>元素的实例。其代码如下所示。

例程 8-7　a.html

```
<body>
链接元素：<a href="http://www.baidu.com">链接内容</a>
</body>
```

其代码运行后，显示效果如图 8-8 所示。当单击页面中含有链接的文本时，显示效果如图 8-9 所示。

注意

此时由于<a>元素中没有定义其他任何属性，所以链接的目标页面将在<a>元素所在的页面中打开，而不会弹出新的页面。

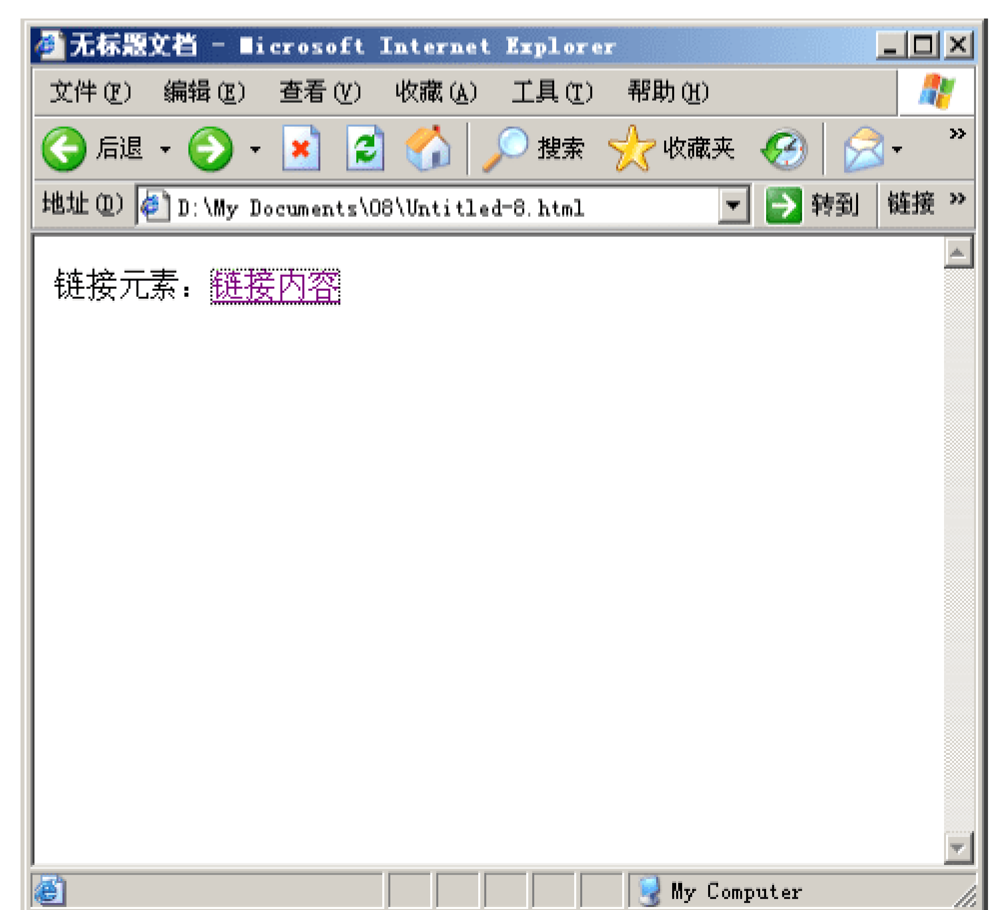


图 8-8　使用<a>元素的显示效果



图 8-9　单击链接文本后的显示效果

<a>元素中可以使用的所有属性如表 8-1 所示。

表 8-1　<a>元素的所有属性

属 性 名 称	写　法
文本显示方向属性	dir
指定语言属性	lang
类属性	class
定义级联样式属性	style
名称属性	name
标题属性	title
标记属性	id
指定路径属性	href
目标文件语言属性	hreflang
指定目标文档类型属性	type
链接快捷键属性	accesskey

(续表)

属性名称	写　法
激活顺序属性	tabindex
指定目标文件编码属性	charset
显示链接目标属性	target
源文档到目标文档关系属性	rel
目标文档到源文档关系属性	rev
敏感区域坐标属性	coords
敏感区域形状属性	shape



说明

其中 coords 属性和 shape 属性，将在讲解<map>元素的使用时具体介绍。

8.2.1　指定路径属性 href

指定路径属性 href 用来指定链接元素<a>的目标地址。从上面路径的知识可以知道，此时使用的路径可以包含 8.1 节中所讲解的所有路径。语法结构如下所示。

```
<a href="链接的路径">链接的文本</a>
```

下面是一个使用 href 属性的实例。其代码如下所示。

例程 8-8　a-href.html

```
<body>
链接元素：<a href="http://www.hao123.com/">链接内容</a>
</body>
```

其代码运行后，显示效果如图 8-10 所示。当单击图 8-10 中含有链接的文本时，显示效果如图 8-11 所示。



图 8-10　使用 href 属性的显示效果



图 8-11　单击链接文本后的显示效果

8.2.2　显示链接目标属性 target

显示链接目标属性 target 用来指定链接目标文件显示的窗口。语法结构如下所示。

```
<a href="链接的路径" target="目标窗口或指定值">链接的文本</a>
```

target 属性中的取值有_self、_blank、_parent 和_top。

下面通过实例演示下每个属性值的表现效果。当 target 属性取值为“_self”时，其代码如下所示。

例程 8-9　a-target.html

```
<body>
链接元素：链接内容<a href="http://www.baidu.com/" target="_self">链接内容</a>
</body>
```

其代码运行后，显示效果如图 8-12 所示。当单击页面中含有链接的文本时，页面跳转，显示效果如图 8-13 所示。

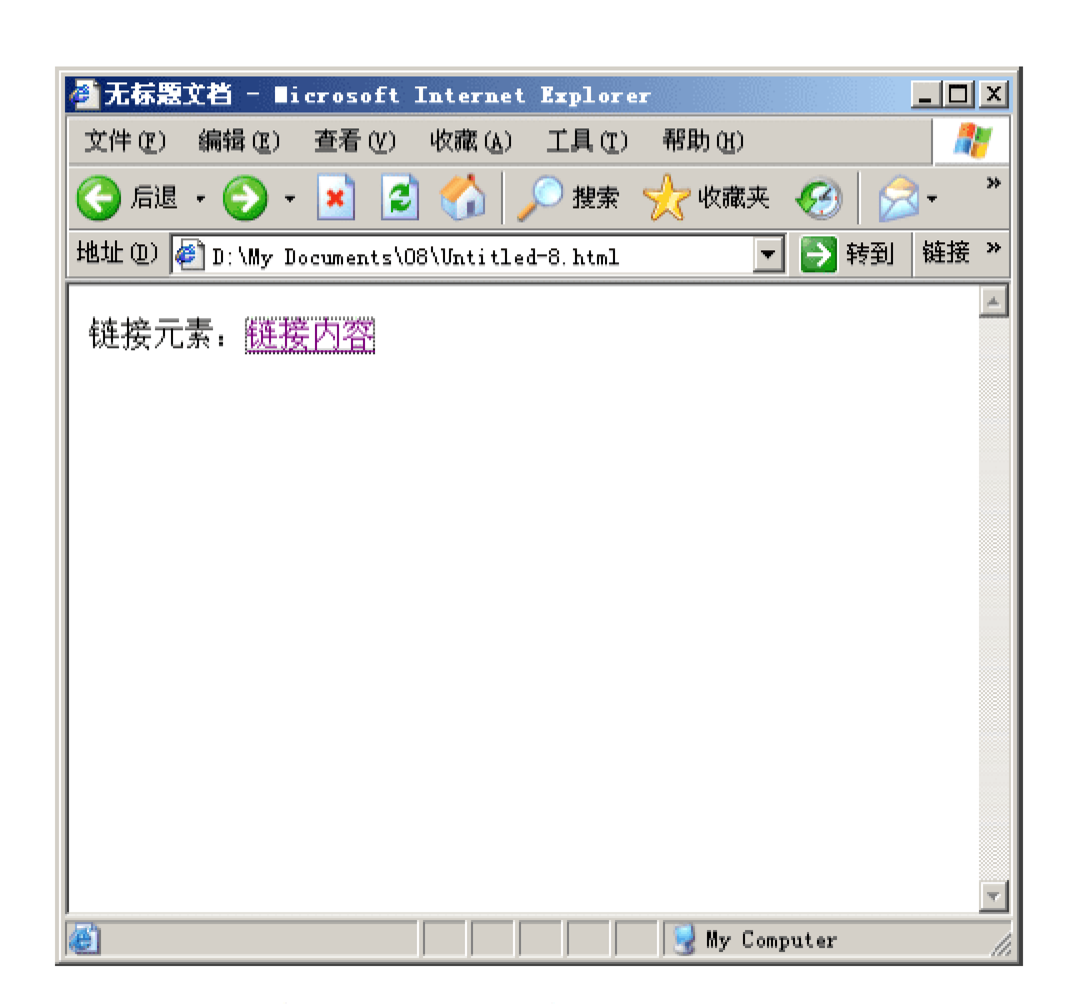


图 8-12　使用 target 属性值为_self 的显示效果



图 8-13　单击链接文本后的显示效果

当 target 属性取值为“_blank”时，其代码如下所示。

例程 8-10　a-targettwo.html

```
<body>
链接元素：链接地址<a href="http://www.baidu.com/" target="_blank">链接地址</a>
</body>
```

其代码运行后，当单击含有链接的内容时，新的内容会在新的窗口显示，而不会覆盖原窗口的内容，其显示效果如图 8-14 所示。

当 target 属性取值为“_parent”和“_top”时，其显示效果和使用“_self”值相同，均为在当前窗口中显示目标页面。

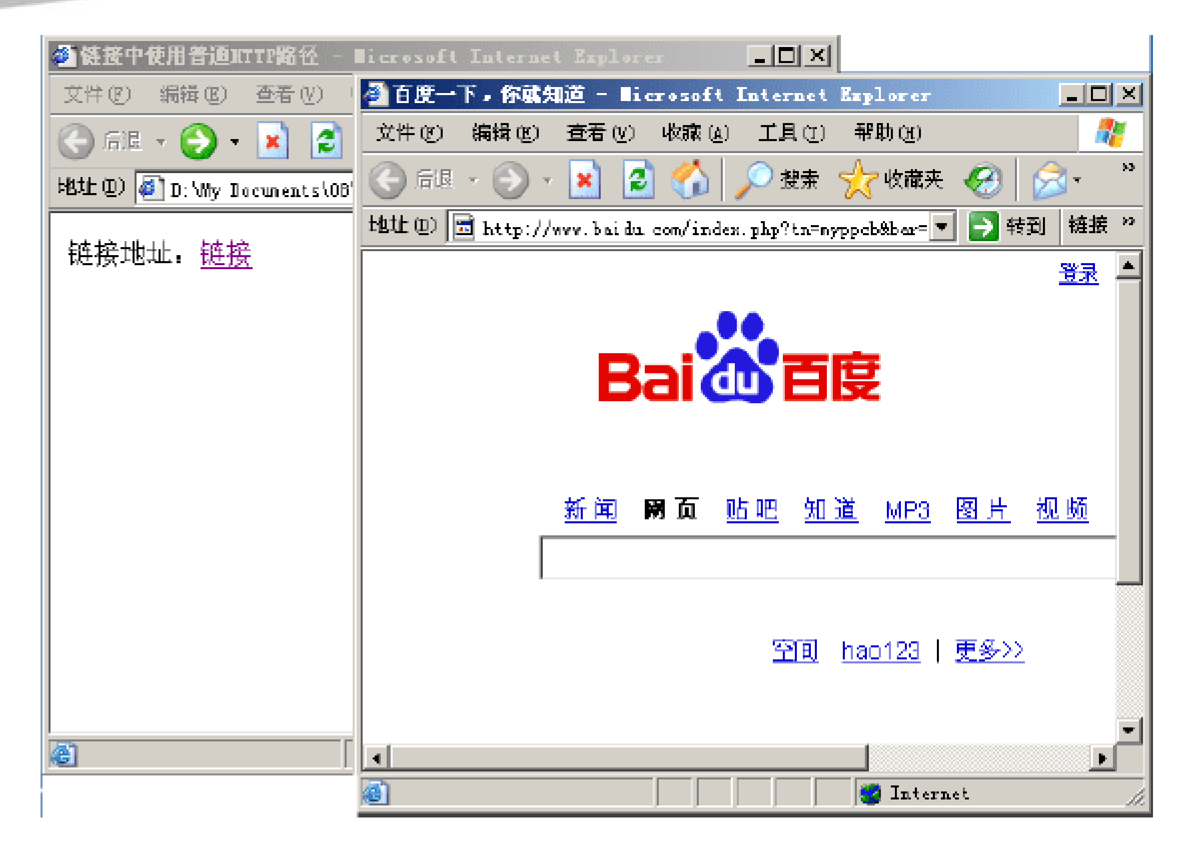


图 8-14　使用 target 属性值为_blank 的显示效果

8.2.3　激活顺序属性 tabindex

激活顺序属性 tabindex 用来指定当按下 Tab 键时，页面中链接的激活顺序。语法结构如下所示。

```
<a href="链接的路径" tabindex="数字值">链接的文本</a>
```

tabindex 属性的取值是大于 0 的整数。下面是一个使用 tabindex 属性的实例。其代码如下所示。

例程 8-11　a-tabindex.html

```
<body>
链接路径 1：<a href="http://www.baidu.com" tabindex="2">链接内容</a><br />
链接路径 2：<a href="http://www.hao123.com" tabindex="3">链接内容</a><br />
链接路径 3：<a href="http://www.sina.com" tabindex="1">链接内容</a>
</body>
```

其代码运行后，按下键盘的 Tab 键可以依次激活页面中的链接。如图 8-15 所示。

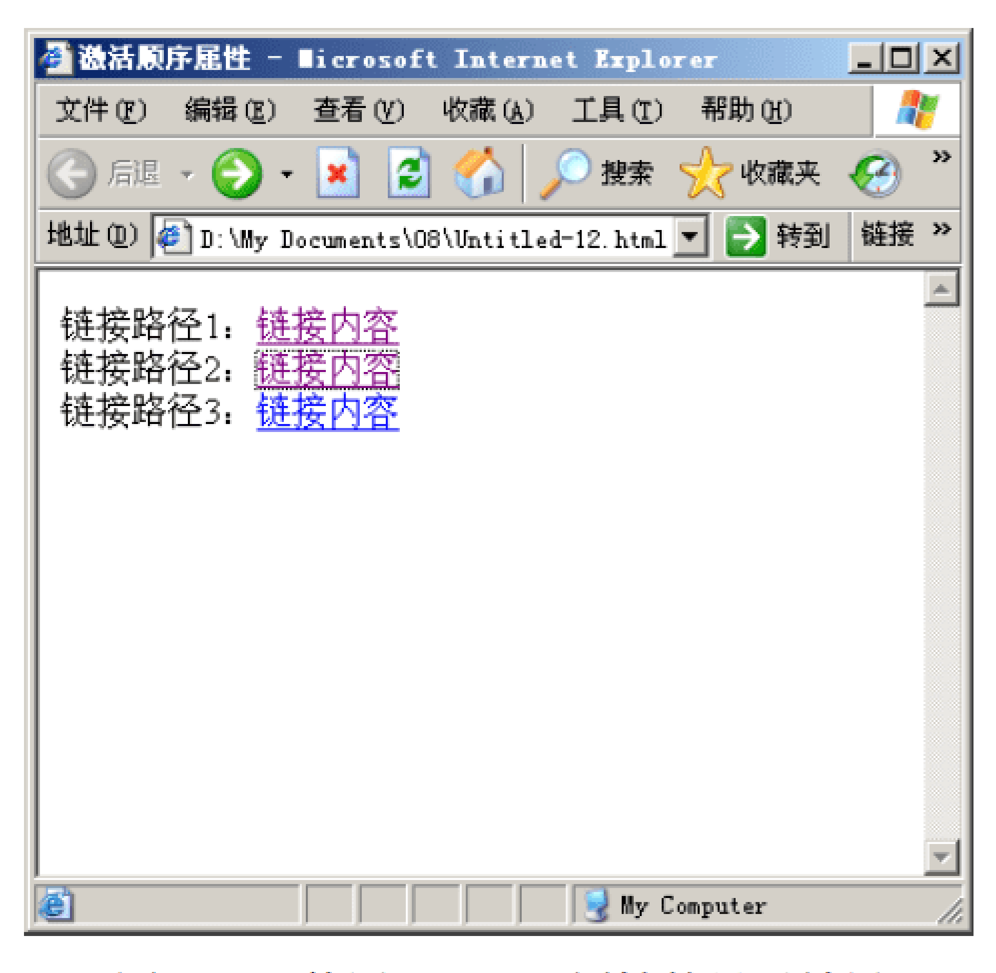


图 8-15　使用 tabindex 属性的显示效果

当按下 Tab 键时，页面中链接元素 2 被激活，如图 8-16 所示。此时，按 Enter 键可以打开链接，其显示效果如图 8-17 所示。

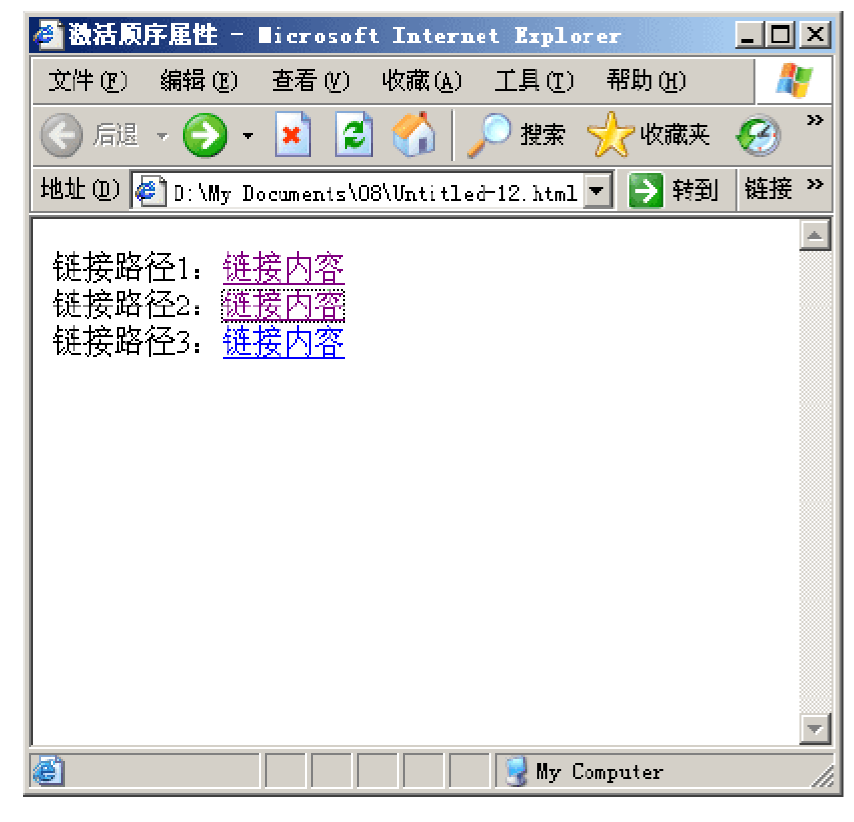


图 8-16　按 Tab 键激活链接后的显示效果



图 8-17　按 Enter 打开链接后的显示效果

8.2.4　链接的热键属性 accesskey

链接的热键属性 accesskey 用来指定激活链接所使用的键。语法结构如下所示。

```
<a href="链接的路径" accesskey="键盘上的键值">链接的文本</a>
```

由于用户使用的操作系统不同，要在键盘上按下相应的键(Window 中为 Alt 键)，同时按下定义的热键，才能激活定义的链接。链接激活后，按下键盘的 Enter 键，打开相应的链接。

下面是一个使用 accesskey 属性的实例。其代码如下所示。

例程 8-12　a-accesskey.html

```
<body>
链接元素：<a href="http://www.google.com" accesskey="a">链接内容</a>
</body>
```

其代码运行后，显示效果，如图 8-18 所示。当同时按下键盘上的 Alt+Z 键时，链接文本被激活，其显示效果如图 8-19 所示。

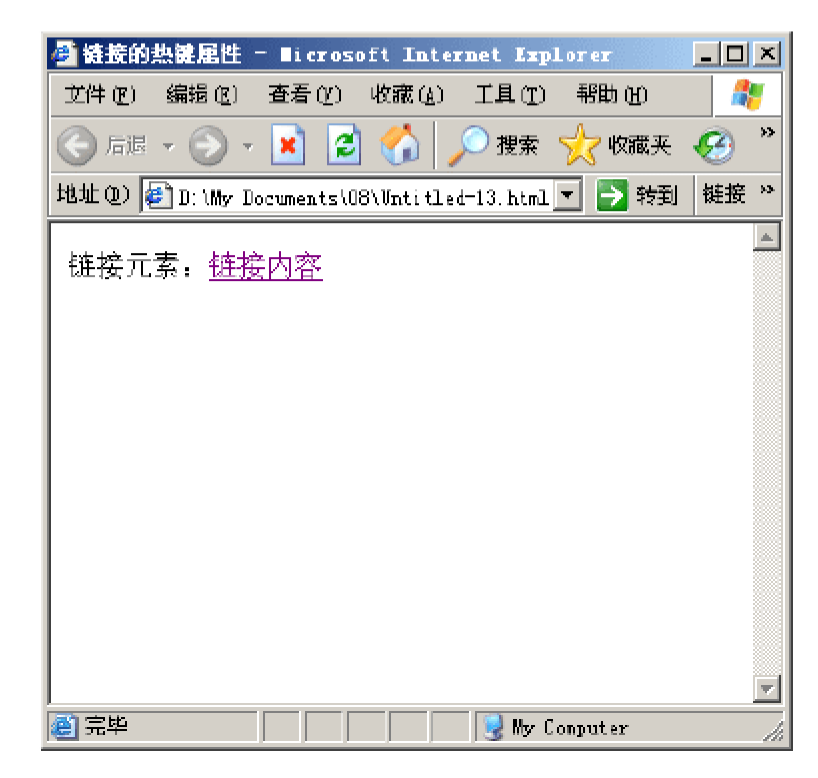


图 8-18　使用 accesskey 属性的显示效果

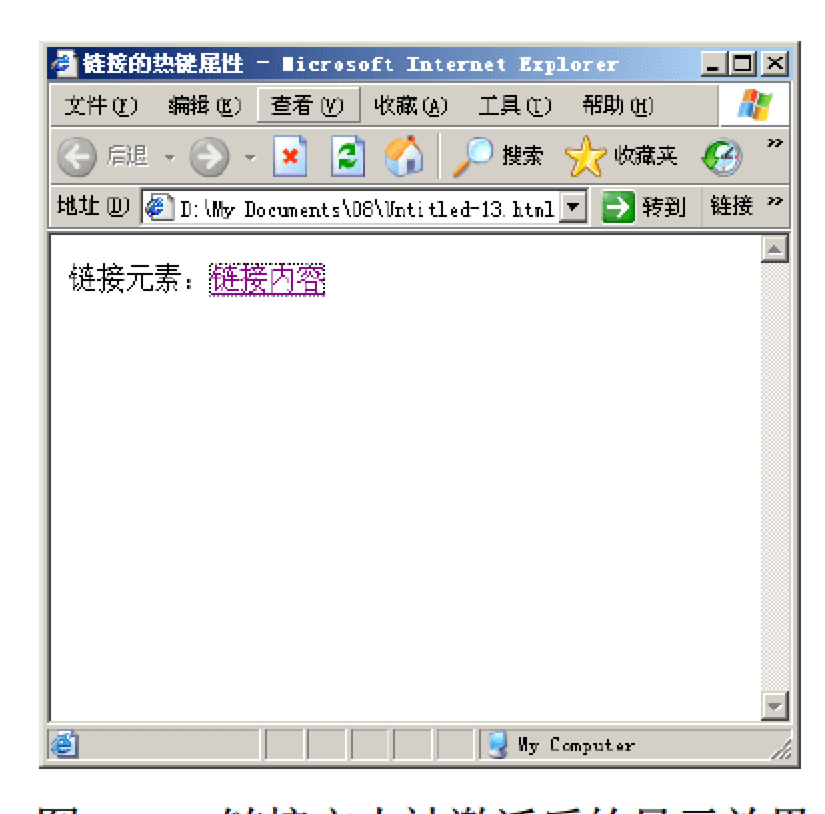


图 8-19　链接文本被激活后的显示效果

从图 8-19 可以看出，激活的链接文本会在文本上产生一个虚线框，用来表明该链接已经被激活。当按下 Enter 键时，可以打开链接的页面，如图 8-20 所示。



图 8-20　按下 Enter 键打开链接后的显示效果

8.3 图像链接

图像中的链接，包括为图像元素制作链接和在图像的局部制作链接。其中在图像的局部制作链接比较复杂，将会使用到<map>、<area>等元素及其相关属性。下面进行详细讲解。

8.3.1　创建链接区域元素<map>

创建链接区域元素<map>用来在图像元素中定义一个链接区域。<map>元素本身并不能指定链接区域的大小和链接目标，<map>元素的主要作用是用来标记链接区域。页面中的图像元素可以使用<map>元素标记的区域。语法结构如下所示。

```
<map>其他元素</map>
```

<map>元素一般要结合<area>元素一起使用，将在讲解<area>元素时(8.3.3 节)再做实例演示。

8.3.2　链接区域的名称属性 name

链接区域的名称属性 name 用来定义链接区域的名称，方便图像元素的调用。语法结构如下所示。



注意

name 属性的取值必须是唯一的。

```
<map name="名称">其他元素</map>
```

name 属性的实例演示，将在 8.3.3 节中讲解。

8.3.3　定义鼠标敏感区元素<area>

定义鼠标敏感区元素<area>用来定义链接区域的大小和坐标，同时可以指定每个敏感区域的链接目标。语法结构如下所示。

```
<map name="名称">
    <area  属性="属性值" />
</map>
```

下面是一个使用<area>元素的实例。其代码如下所示。

例程 8-13　area.html

```
<body>
<img src="pic.jpg" alt="雷峰塔" border="0" usemap="#Map" />
<map name="Map" id="Map">
<area shape="rect" coords="27,20,132,122" href="http://www.baidu.com" />
<area shape="rect" coords="226,61,322,200" href="http://www.hao123.com" />
<area shape="circle" coords="114,235,52" href="http://www.sina.com" />
</map>
</body>
```

该实例中，在图片中定义了 3 个链接区域，分别链接到百度、谷歌和 W3C 的站点首页。在图片的局部制作链接后，对图片的显示效果并没有影响。其代码运行后，显示效果如图 8-21 所示。

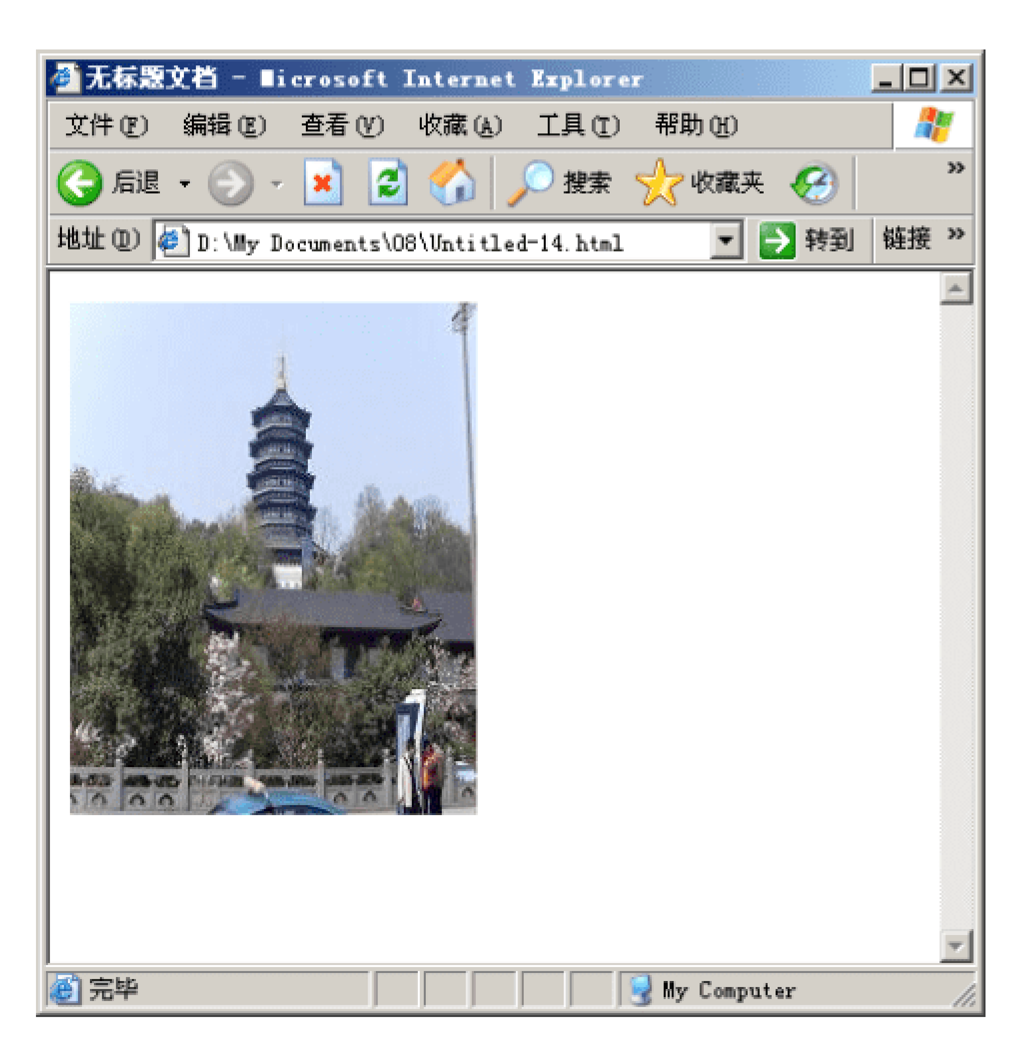


图 8-21　局部定义链接后的图像元素

<area>元素中可以使用的所有属性如表 8-2 所示。

表 8-2　<area>元素的所有属性

属 性 名 称	写　法
文本显示方向属性	dir
指定语言属性	lang
类属性	class
定义级联样式属性	style
名称属性	name
标题属性	title
标记属性	id
指定路径属性	href
不指定路径属性	nohref
链接的文本说明属性	alt
链接快捷键属性	accesskey
激活顺序属性	tabindex
敏感区域坐标属性	coords
敏感区域形状属性	shape

8.3.4　链接的路径属性 href、nohref

指定路径属性 href 用来指定链接的地址。不指定路径属性 nohref 用来不指定链接的地址。其中 href 属性和<a>元素中完全相同。使用指定路径属性 href 的语法如下所示。

```
<map name="名称">
    <area href="指定的路径" />
</map>
```

不指定路径属性 nohref 的语法如下所示。

```
<map name="名称">
    <area nohref="nohref" />
</map>
```

下面是一个在<area>元素中使用 nohref 属性的实例。其代码如下所示。

例程 8-14　area-href.html

```
<body>
<img src="pic.jpg" alt="雷峰塔" border="0" usemap="#Map" />
<map name="Map" id="Map">
<area shape="rect" coords="27,20,132,122" href="http://www.baidu.com" />
<area shape="rect" coords="226,61,322,200" href="http://www.hao123.com" />
<area shape="circle" coords="114,235,52" href="http://www.sina.com" />
</map>
```

```
</body>
```

使用了 nohref 属性的区域，当鼠标悬停时，将不再显示“手”的形状。

8.3.5 链接的文本说明属性 alt

链接的文本说明属性 alt 用来使用附加的文本对链接进行说明。和<img>元素里的 alt 属性类似，当鼠标悬停在某个链接区域时，将会显示出 alt 属性中附加的文本。语法结构如下所示。

```
<map name="名称">
    <area alt="附加的文本" />
</map>
```

下面是一个在<area>元素中使用 alt 属性的实例。其代码如下所示。

例程 8-15 area-alt.html

```
<body>
<img src="pic.jpg" alt="雷峰塔" border="0" usemap="#Map" />
<map name="Map" id="Map">
<area shape="rect" coords="30,21,132,112" href="http://www.baidu.com" alt="左侧链接百度" />
<area shape="rect" coords="221,62,312,210" href="http://www.hao123.com" alt="右侧链接 hao123" />
<area shape="circle" coords="104,245,54" href="http://www.sina.com" alt="底部链接新浪" />
</map>
</body>
```

其代码运行后，当鼠标悬停在相应区域时，会出现相应的链接。显示效果如图 8-22 所示。

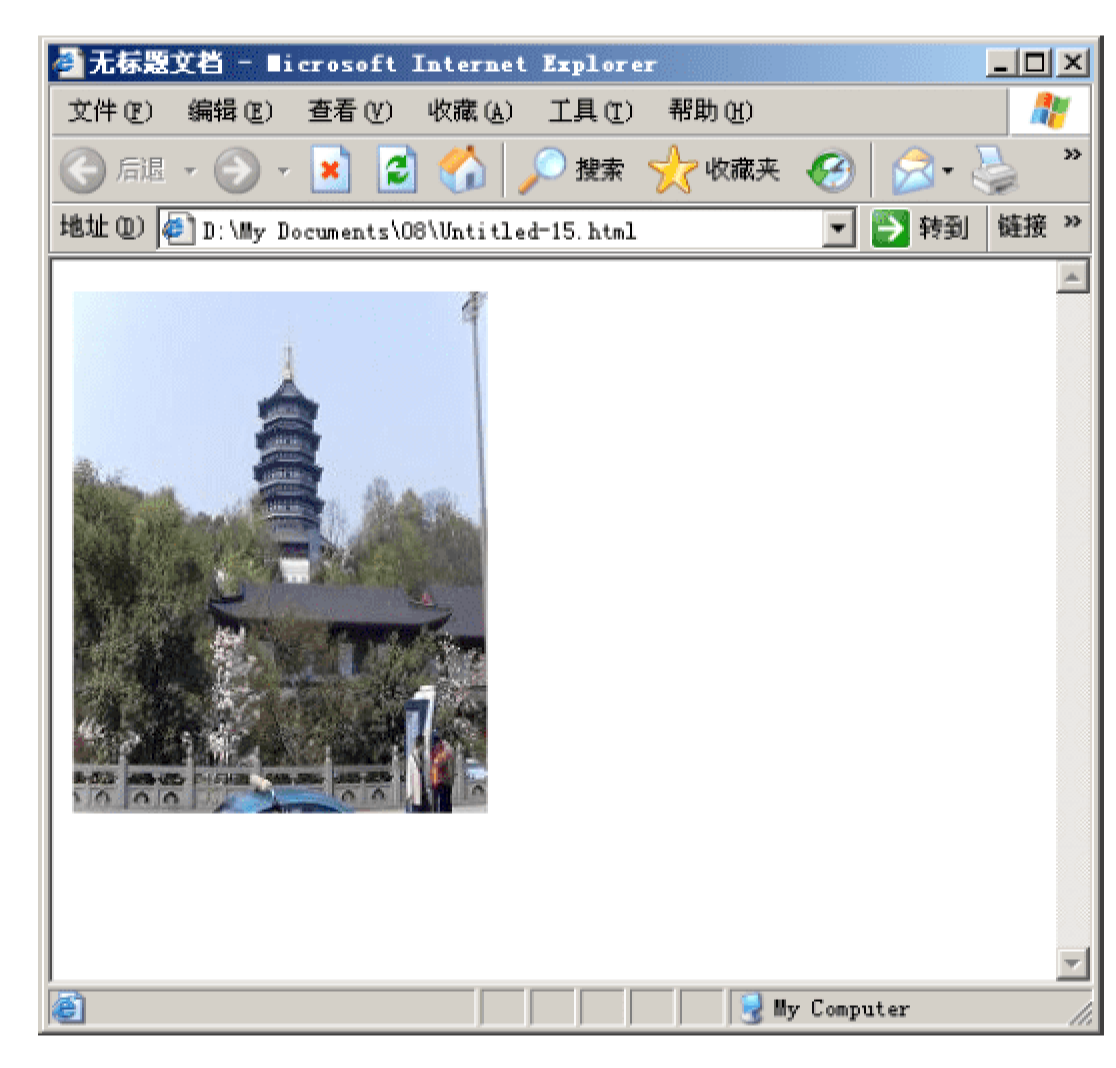


图 8-22 <area>元素中使用 alt 属性的图像元素

8.3.6 鼠标敏感区坐标属性 coords

鼠标敏感区坐标属性 coords 用来定义鼠标敏感区域的大小和位置。根据敏感区域的形状不同，所使用的坐标数目也会有所变化。语法结构如下所示。

```
<map name="名称">
    <area coords="区域坐标组" />
</map>
```

对应不同形状的敏感区域，其坐标的具体定义方法如下所示。

1. 矩形区域

定义一个矩形区域要使用 4 个坐标来实现，其形式如下所示。

```
coords="x1,y1,x2,y2"
```

每个坐标之间用英文的逗号分隔，其中 x1、y1 表示矩形区域左上角的坐标，x2、y2 表示矩形区域右下角的坐标。图片的左上角是坐标的原点，其坐标为“0,0”。

2．圆形区域

定义一个圆形区域要使用 3 个坐标来实现，其形式如下所示。

```
coords="x,y,r"
```

每个坐标之间用英文的逗号分隔，其中 x、y 坐标表示圆形区域圆心的坐标，r 表示圆形区域的半径的长度。

3．多边形区域

定义一个多边形区域要使用和顶点数目相同的坐标组来实现，其形式如下所示。

```
coords="x1,y1, x2,y2…"
```

每个坐标之间用英文的逗号分隔，其中每组 x、y 坐标表示多边形区域的一个顶点。

下面是一个在<area>元素使用 coords 属性的实例。其代码如下所示。

例程 8-16 area-coords.html

```
<body>
<img src="pic.jpg" alt="雷峰塔" border="0" usemap="#Map" />
<map name="Map" id="Map">
<area shape="circle" coords="154,221,,62" href="http://www.baidu.com" alt="左侧链接百度" />
<area shape="rect" coords="221,62,312,210" href="http://www.hao123.com" alt="右侧链接 hao123" />
<area shape="rect" coords="24,25,123,134" href="http://www.sina.com" alt="底部链接新浪" />
</map>
</body>
```

其代码运行后，按下 Tab 键，可以激活链接区域，其中第一个圆形区域的显示效果如图 8-23 所示。按照同样的方法，激活后的矩形区域的显示效果如图 8-24 所示。

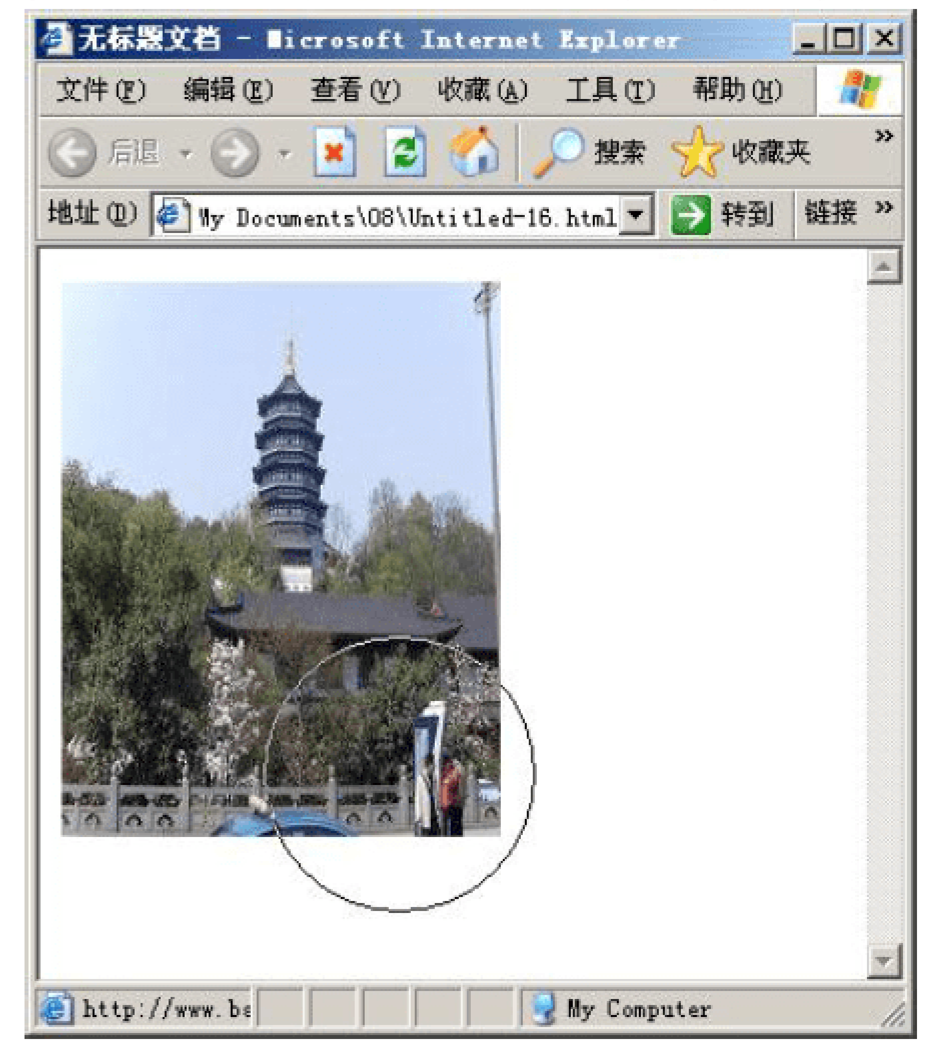


图 8-23 <area>元素中使用 coords 属性的显示效果 1

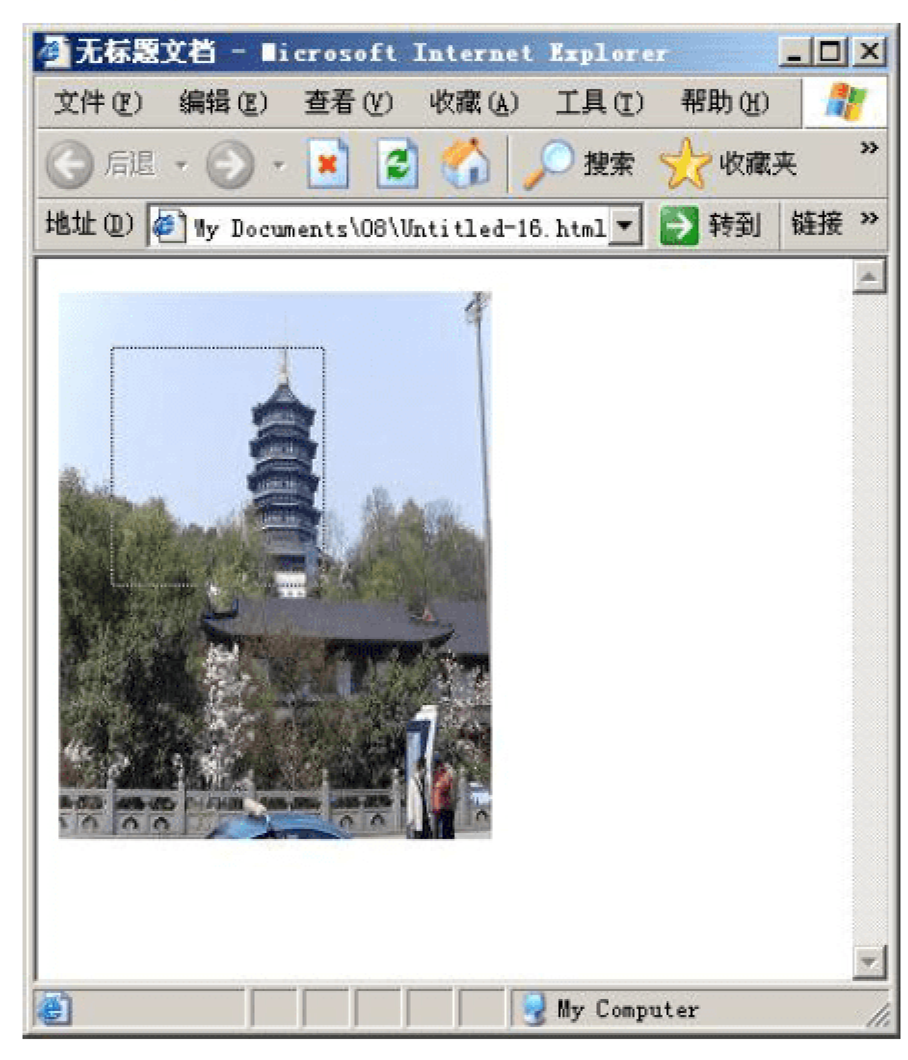


图 8-24 <area>元素中使用 coords 属性的显示效果 2

8.3.7 鼠标敏感区形状属性 shape

鼠标敏感区形状属性 shape 用来定义鼠标敏感区域形状。其具体的大小和位置要使用 coords 属性定义。语法结构如下所示。

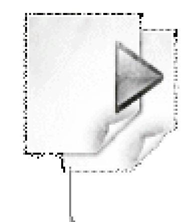
```
<map name="名称">
    <area shape="形状" />
</map>
```

shape 属性的显示效果，8.3.6 节的实例 8-16 已经用过，读者可以回顾 8.3.6 节所学内容。

8.3.8 使用图片中的链接

综合使用<img>元素、<map>元素和<area>元素制作一个图片局部的链接。语法结构如下所示。

```
<img src="图片路径" usemap="map 中定义的 name 或 id" />
<map name="名称" id="标记">
    <area href="路径" cords="坐标组" shape="形状" />
</map>
```



注意

其中<img>元素中 usemap 属性的取值中，要在具体值前加“#”号。

下面是一个制作图片局部链接的实例。其代码如下所示。

例程 8-17 img-use-link.html

```
<body>
<img src="pic.jpg" alt="老虎" border="0" usemap="#Map" />
<map name="Map" id="Map">
<area shape="rect" coords="27,20,132,122" href="http://www.baidu.com" alt="链接 1" target="_blank" />
<area shape="rect" coords="226,61,322,200" href="http://www.google.com" alt="链接 2" />
<area shape="circle" coords="114,235,52" href="http://www.w3.org" alt="链接 3" />
</map>
</body>
```

其代码运行后，单击图片中出现的圆圈后会链接到百度首页，显示效果如图 8-25 所示。

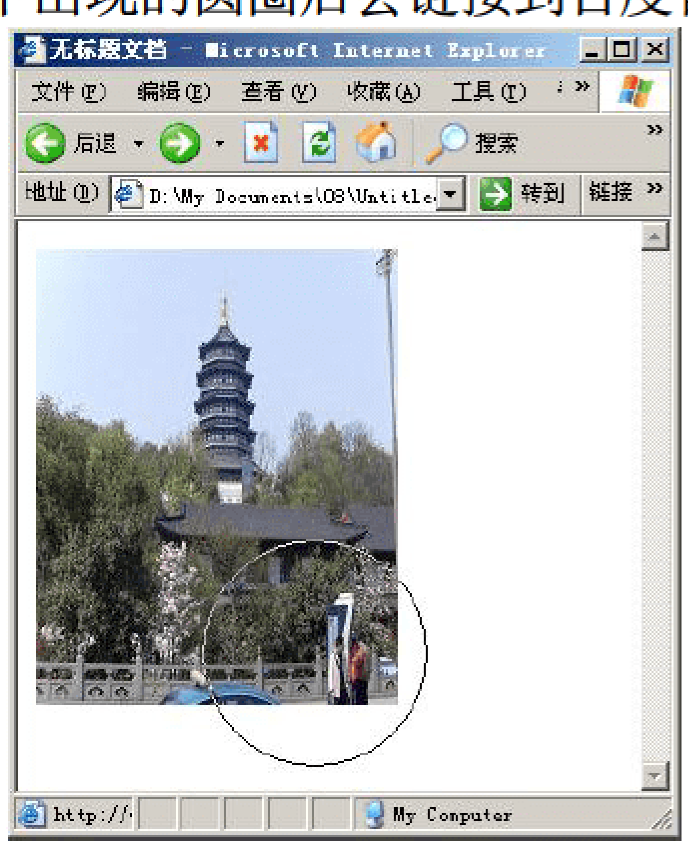


图 8-25 图片局部链接的显示效果

8.4 本章习题

一、选择题

1．在网页中，必须使用(　　)标记来完成超级链接。

A. <a>…　　B. <p>…</p>　　C. <link>…</link>　　D. …

2．若要在网页中插入样式表 main.css，以下用法中，正确的是(　　)。

A. <Link href=“main.css”type=text/css rel=stylesheet>

B. <Link Src=“main.css”type=text/css rel=stylesheet>

C. <Link href=“main.css”type=text/css>

D. <Include href=“main.css”type=text/css rel=stylesheet>

二、填空题

1. 如果要创建一个指向电子邮件 someone@mail.com 的超链接，代码应该如下：_____。

2. 在给图像指定超链接时，默认情况下总是会显示蓝色边框，如果不想显示蓝色边框，应使用以下语句：________。

3．在指定页内超链接时，如果在某一个位置使用了<a ________="target1">锚点语句定义了锚点，那么应使用以下语句，以便在单击超链接时跳转到锚点定义的位置：<a href=________>。

4. 创建超链接时，要使目标文件在一个新窗口中打开，应使用<a href=test.htm________>。

5. 设置超链接的目标框架时，除了可以使用特殊框架名，还可以使用框架集文件中标记符指定的 name 属性所对应的框架。

三、实战练习

1．使用激活顺序属性 tabindex 设置一个页面，链接 3 个不同的网址。

2．使用鼠标敏感区元素<area>，用 1 个图像链接 3 个不同的网址。

Chapter 9

表单元素

表单元素是页面中用来提供用户交互的主要方法。一般将表单设计在一个 HTML 文档中，当用户填写完信息后做提交操作，于是表单的内容就从客户端的浏览器传送到服务器上，经过服务器上处理程序处理后，再将用户所需信息传送回客户端的浏览器上，这样网页就具有了交互性。

在网页中，最常见的表单形式主要包括文本框、单选按钮、复选项、按钮等。如图 9-1 所示，在新浪的主页中，就包含了文本框、按钮、下拉列表等表单内容。



图 9-1　新浪首页

本章主要内容有:

◎　熟练掌握表单元素的各个属性。

◎　重点掌握表单中的按钮属性。

◎　能够自由应用表单元素的各个属性并能制作相应的表单。

9.1 表单元素<form>

在 HTML 中，<form></form>标志对用来创建一个表单，也即定义表单的开始和结束位置，在标志对之间的一切都属于表单的内容。<form>元素可以在页面中任何位置使用，当其中没有任何控件时，本身并没有表现效果。语法结构如下所示。

```
<form>……</form>
```

注意

在网页中，<form>元素一般要包含至少一个表单控件，每个表单元素开始于 form 元素，可以包含所有的表单控件，还有任何必需的伴随数据，如控件的标签、处理数据的脚本或程序的位置等。在表单的<form>标记中还可以设置表单的基本属性，包括表单的名称、处理程序、传送方法等。一般情况下，表单的处理程序 action 和传送方法 method 是必不可少的参数。

下面是一个使用<form>元素的实例。其代码如下所示。

例程 9-1 form.html

```
01 <body>
02 <form action="#" method="post">
03 <div id="form1">
04 <h3>注册信息</h3>
    用户姓名：
05 <input type="text" value="请填写姓名" size="20" name="name1" /><br />
   个人说明：
06 <input type="text" value="请用 4 个字概括自己" size="20" name="name2" /><br />
   个性签名：
07 <textarea name="name3" cols="40" rows="10"></textarea>
08 <p>写出自己的心声</p>
09 <div>
10 </form>
11 </body>
```

该实例中，05 和 07 行中分别使用了<input>元素和<textarea>元素。其运行后的显示效果如图 9-2 所示。

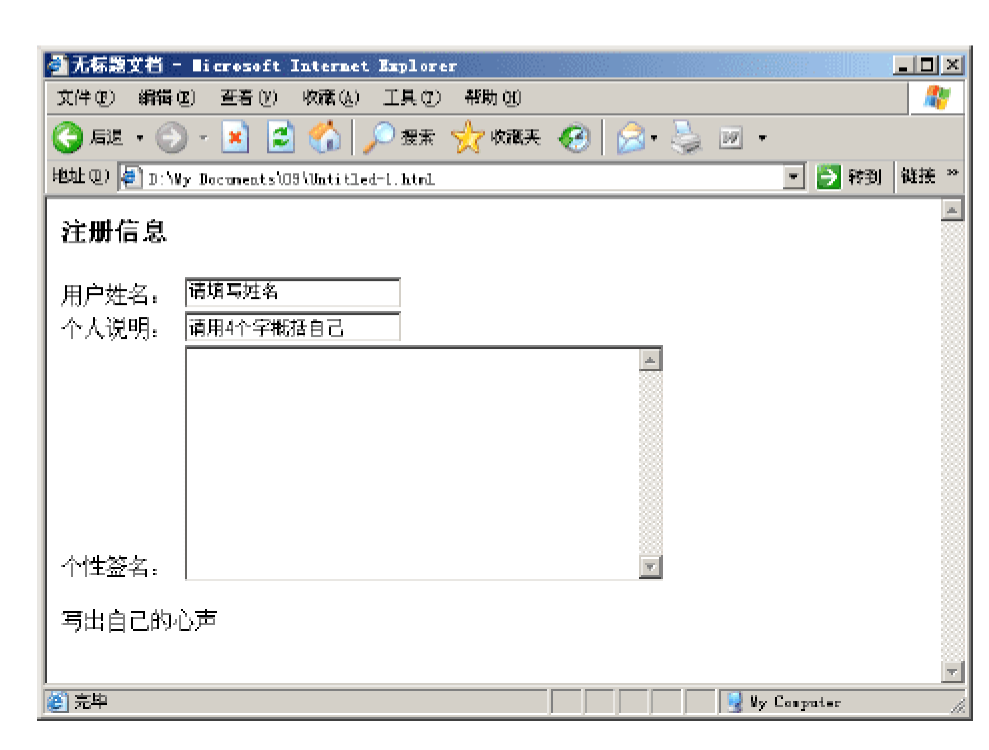
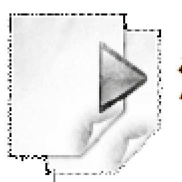


图 9-2　使用<form>元素的显示效果

9.1.1　动作属性 action

动作属性 action 用来指定处理表单的程序的路径。一般情况下，action 属性主要用来处理用户通过表单提交的信息，如保存、回复等。



注意

action 属性是<form>元素中必须定义的属性。

语法结构如下所示。

```
<form action="  表单的处理程序">其他元素</form>
```

在该语法中，表单的处理程序定义的是表单要提交的地址，也就是表单中收集到的资料将要传递的程序地址。这一地址可以是绝对地址，也可以是相对地址，还可以是一些其他的地址形式，如发送 E-mail 等。下面是一个使用 action 属性的实例，其代码如下所示。

例程 9-2 form-action.html

```
<body>
<form action="mailto:abcd@163.com">
<div id="form1">
<h3>注册信息</h3>
 用户姓名：
<input type="text" value="请填写姓名" size="20" name="name1" /><br />
  个人说明：
<input type="text" value="请用 4 个字概括自己" size="20" name="name2" /><br />
  个性签名：
<textarea name="name3" cols="40" rows="10"></textarea>
<p>写出自己的心声</p>
<div>
</form>
</body>
```

该实例中，指定处理回复信息的程序为 feedback.asp。当单击“提交”按钮后，程序将处理提交的信息，并返回相应的内容。由于服务器端程序，不是本书讲解的内容，所以就不讲解程序的处理方式和返回内容了。在<form>元素中使用 action 属性，并不影响<form>元素中内容的显示效果。其代码运行后，显示效果如图 9-3 所示。

图 9-3 <form>元素中使用 action 属性的显示效果

9.1.2 发送数据方式属性 method

表单的 method 属性用来定义处理程序从表单中获得信息的方式，可取值为 get 和 post 的其中一个，它决定了表单中已收集数据是用什么样的方法发送到服务器的。下面分别作详细讲解。

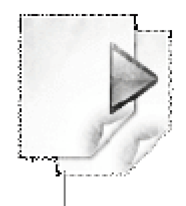
- method=get：使用这个设置时，表单数据会被视为 CGI 或 ASP 的参数发送，也就是来访者输入的数据会附加在 URL 之后，由用户端直接发送至服务器，所以速度上会比 post

快，但缺点是数据长度不能够太长。在没有指定 method 的情形下一般都会视 get 为默认值。

- method=post：使用这种设置时，表单数据是与 URL 分开发送的，用户端的计算机会通知服务器来读取数据，所以通常没有数据长度上的限制，缺点是速度上也会比 get 慢。

语法结构如下：

```
<form method="传送方式">
……</form>
```



说明

传送方式的值只有 get 或 post 两种选择。

下面是一个使用 method 属性的实例。其代码如下所示。

例程 9-3 form-mothod.html

```
<body>
<h3>一个对图书馆服务回复的示例</h3>
<form action="mailto:sucess111@163.com" name="research" method="post">
用户姓名：
<input type="text" value="请填写姓名" size="30" name="name1" /><br />
用户反馈：
<textarea cols="40" rows="10" name="name1"></textarea><br />
<input type="submit" value="提交" />
</form>
</body>
```

其代码运行后，显示效果(submit 属性将在 9.2.3 节介绍)如图 9-4 所示。单击“提交”按钮后，其显示效果如图 9-5 所示。

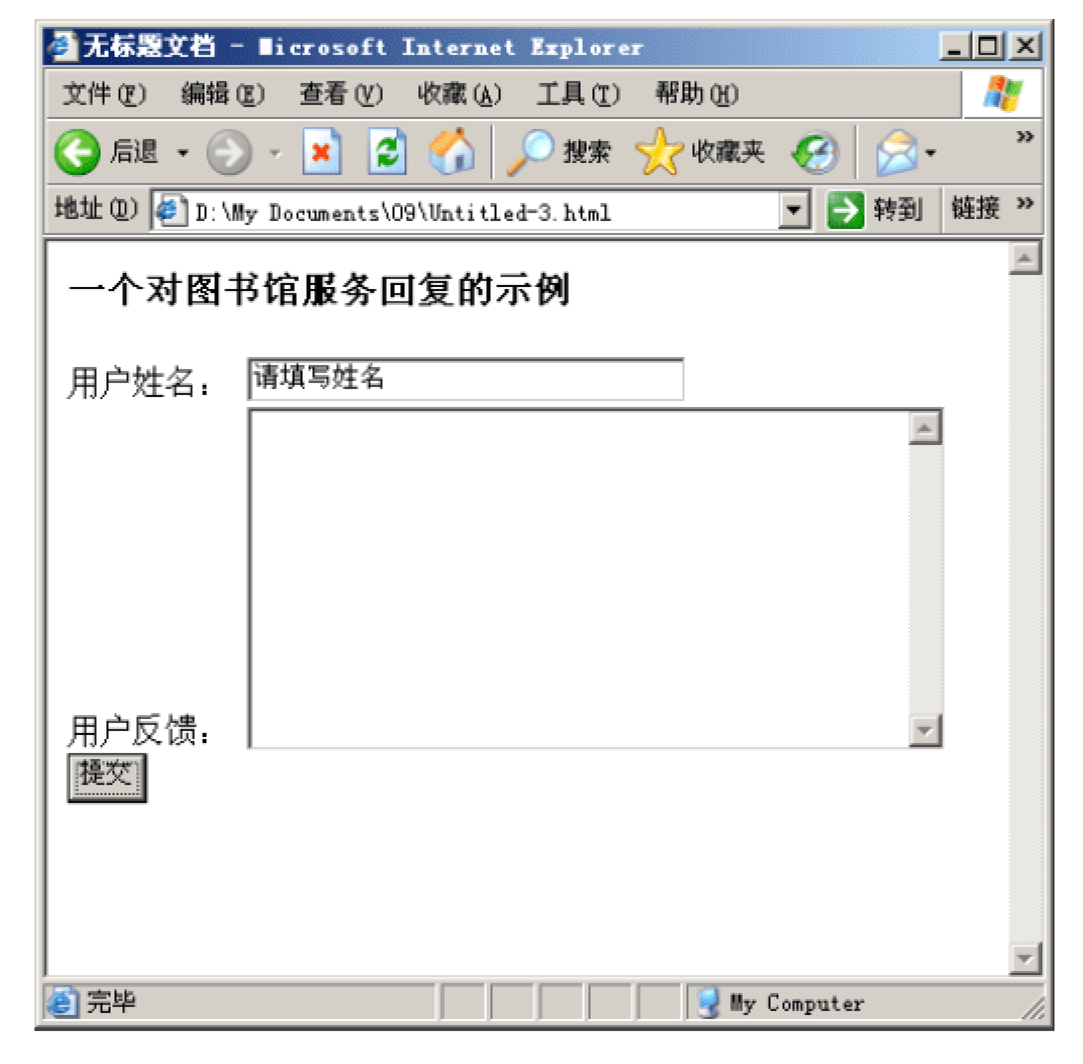


图 9-4　<form>元素中使用 method 属性的显示效果

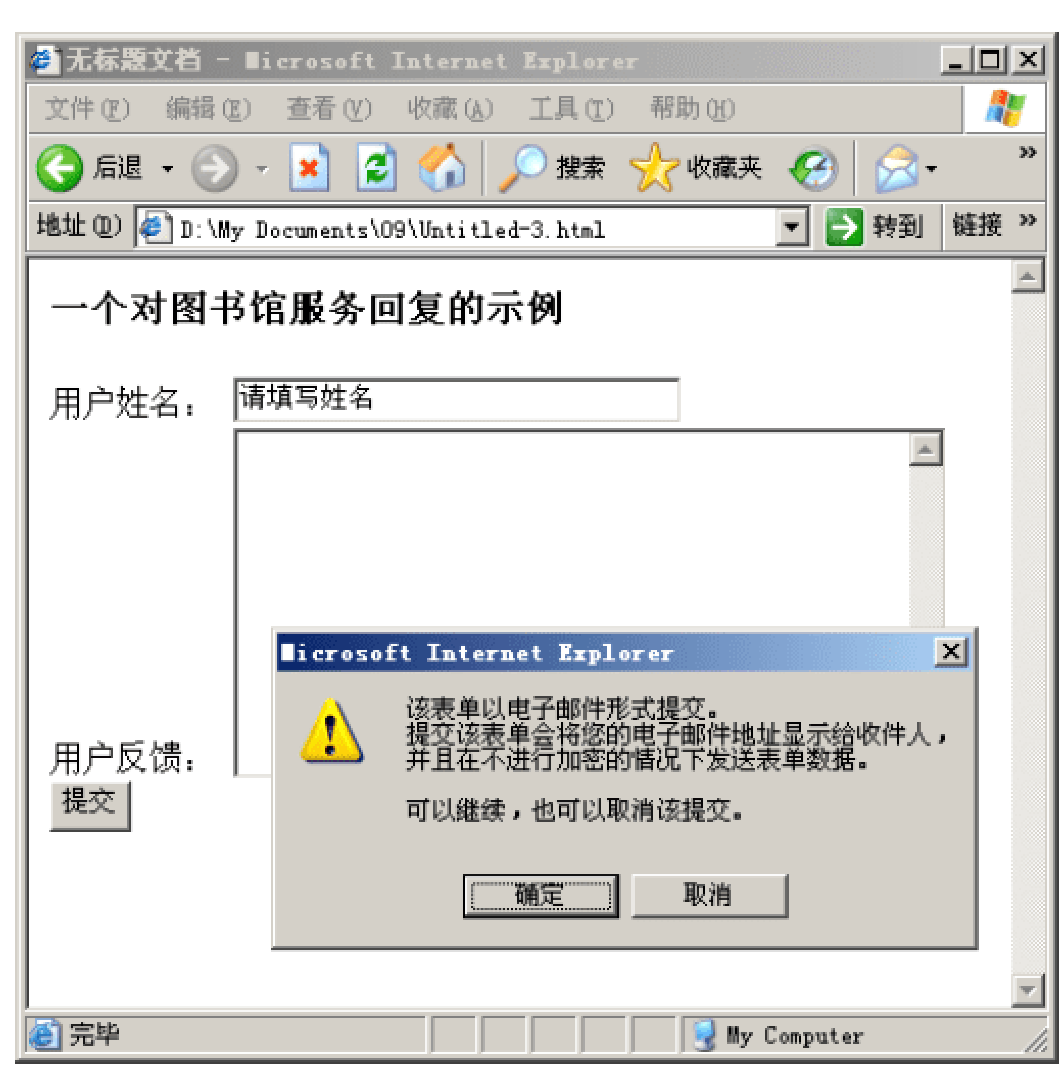


图 9-5　提交数据后的显示效果

9.1.3 名称属性 name

表单名称属性 name 用来指定表单的名称，标记某个表单，方便程序的处理。使用 id 属性也可以达到相同的效果，为了兼容 Netspace，最好同时定义 name 和 id 属性。语法结构如下所示。

```
<form action="程序路径" name="名称">其他元素</form>
```

下面是一个使用 name 属性的实例。其代码如下所示。

例程 9-4　form-name.html

```
<body>
<h3>一个对图书馆服务回复的示例</h3>
<form action="mailto:abcd@163.com" name="research" method="post">
用户姓名：
<input type="text" value="请填写姓名" size="30" name="name1" /><br />
用户反馈：
<textarea cols="40" rows="10" name="name1"></textarea><br />
<input type="submit" value="提交" />
</form>
</body>
```

使用 name 属性，对表单的显示效果并没有任何影响。

9.2 表单控件<input>

表单控件元素<input>用来在页面中定义输入内容(或者进行选择)的部分。其中根据表现效果的不同，可以分为文本框、复选框、单选按钮等。语法结构如下所示。

```
<input  属性="属性值’ />
```

在默认情况下，<input>元素会同文本、图片等元素同行显示。

9.2.1 文本域 text

文本域中<input>元素的 type 属性值为 text，其主要作用是用来提供用户输入文本的功能。文本域只能单行显示。超出定义宽度的内容将被隐藏。语法结构如下所示。

```
<input type="text"  属性="属性值" />
```

下面是一个使用文本域的实例。其代码如下所示。

例程 9-5　input-text.html

```
<body>
<h3>个人信息</h3>
<form action="edit.asp">
```

```
姓名：
<input type="text" size="30" name="1" /><br />
性别：
<input type="text" size="30" name="2" /><br />
民族：
<input type="text" size="30" name="3" /><br />
年龄：
<input type="text" size="30" name="4" /><br />
籍贯：
<input type="text" size="30" name="5" /><br />
<input type="submit" value="提交" />
</form>
</body>
```

其运行后的显示效果如图 9-6 所示。



图 9-6 文本域的显示效果

9.2.2 密码区域 password

密码区域中<input>元素的 type 属性值为 password，其主要作用是用来提供输入密码、口令等内容的区域。该区域中输入的内容将以实心圆点的形式显示。语法结构如下所示。

```
<input type=" password" 属性="属性值" />
```

下面是一个使用密码区域的实例。其代码如下所示。

例程 9-6 input-password.html

```
<body>
下面是几种不同效果的密码域：
```

```
<form name="example" action="deal.asp" method="post">
    <!--添加一个长度为 22 的密码域-->
    登录密码：<input type="password" name="username" size=22><br>
    <!--添加一个长度为 22，但是最多可以输入 30 个字符的密码域-->
    支付密码：<input type="password" name="age" size=22 maxlength=30><br>
    <!--添加一个长度为 22、最多可输入 30 个字符，默认密码设置为 12345 的密码域-->
    原始密码：<input type="password" name="privateweb" size=22 maxlength=30 value="12345">
</form>
</body>
```

其运行后，添加账号和密码时的显示效果，如图 9-7 所示。在页面中输入任意字符后的显示效果如图 9-8 所示。

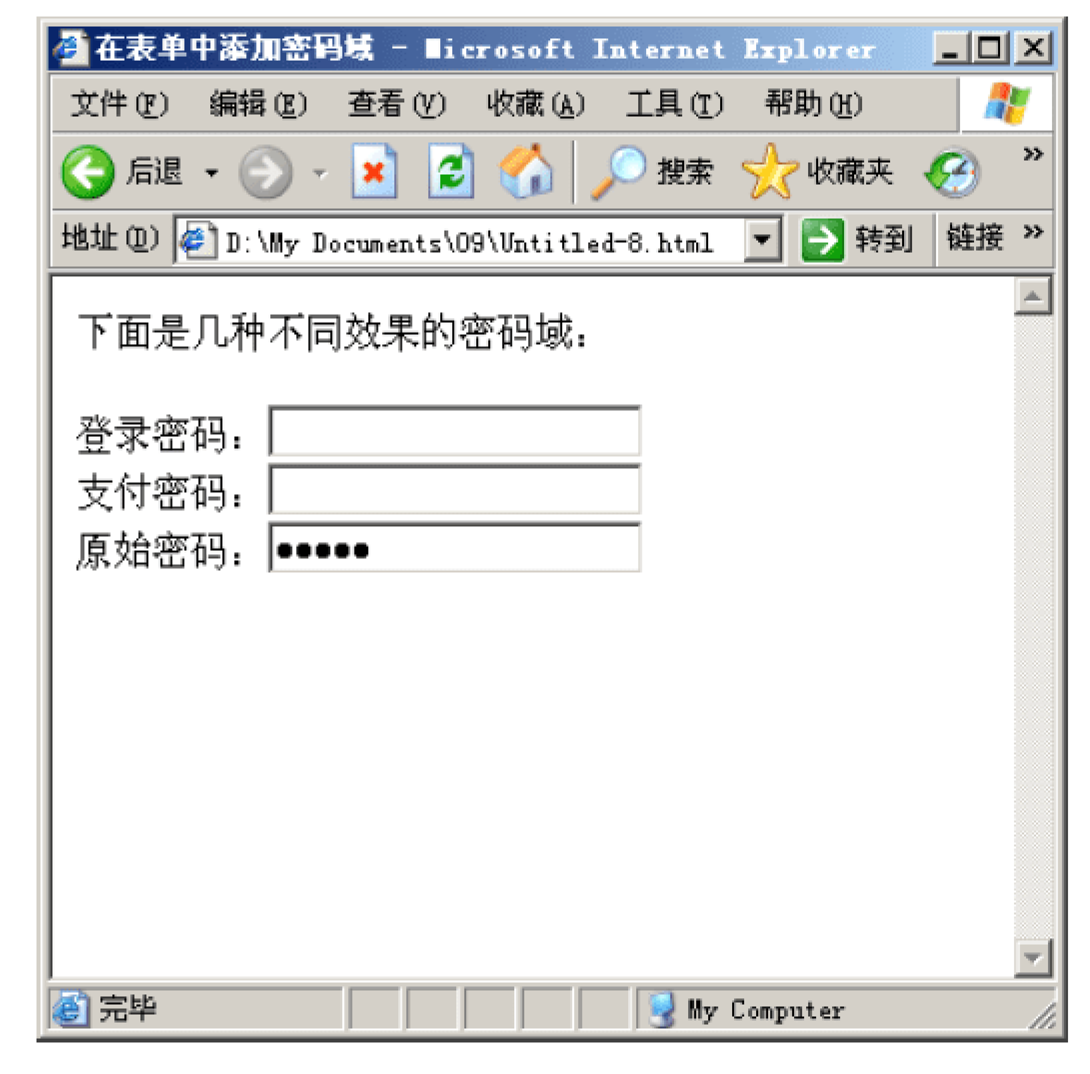



图 9-7 密码区域的显示效果



图 9-8 在密码域输入文字

9.2.3 提交按钮 submit

提交按钮中<input>元素的 type 属性值为 submit，用来发送表单中的内容。只有单击“提交”按钮，表单中的信息才能够发送给相应的程序。语法结构如下所示。

```
<input type=" submit"  属性="属性值" />
```

下面是一个使用提交按钮的实例。其代码如下所示。

例程 9-7 input-submit.html

```
<body>
<h3>网银支付</h3>
<form action="edit.asp">
用户账号：<input type="text" size="20" name="name1" /> <br />
登录密码：<input type="password" name="username" size=30><br>
支付密码：<input type="password" name="age" size=30 ><br>
<input type="submit" name="name3" value="确认提交" />
```

```
</form>
</body>
```

其运行后的显示效果，如图 9-9 所示。



图 9-9　提交按钮的显示效果

9.2.4　复位按钮 reset

复位按钮中<input>元素的 type 属性值为 reset，其主要作用是，当用户想重新填写内容时，可以使用复位按钮，使表单恢复到初始状态。语法结构如下所示。

```
<input type=" reset"  属性="属性值" />
```

下面是一个使用复位按钮的实例。其代码如下所示。

例程 9-8　input-reset.html

```
<body>
<h3>网银支付</h3>
<form action="edit.asp">
  用户账号：<input type="text" size="20" name="name1" /><br />
  登录密码：<input type="password" name="username" size=30><br>
   支付密码：<input type="password" name="age" size=30 ><br>
<input type="submit" name="name3" value="确认提交" />
<input type="reset" name="name4" value="复位" />
</form>
</body>
```

其运行后的显示效果，如图 9-10 所示。

从图 9-10 可以看出，复位按钮和提交按钮在显示效果上并没有区别。所以要注意 value 属性中文本要正确对应。



图 9-10　复位按钮的显示效果

9.2.5　图像按钮 image

图像按钮中<input>元素的 type 属性值为 image，用来发送表单中的内容。其作用和提交按钮的作用相同。语法结构如下所示。

```
<input type=" image"  属性="属性值" />
```

注意

在图像按钮中，可以使用图像元素中的 src 属性定义图像的路径。用 width 和 height 属性定义图片的大小(建议使用级联样式表控制图片的表现)。

下面是一个使用图像按钮的实例。其代码如下所示。

例程 9-9　input-image.html

```
<body>
<h3>网银支付</h3>
<form action="edit.asp">
用户账号：<input type="text" size="20" name="name1" /><br />
登录密码：<input type="password" name="username" size=30><br>
支付密码：<input type="password" name="age" size=30 ><br>
<input type="image" src="http://www.baidu.com/img/logo.gif" width="90" height="40" name="name3"
value="提交" />
<input type="reset" name="name4" value="复位" />
</form>
</body>
```

其运行后的显示效果，如图 9-11 所示。



图 9-11　图像按钮的显示效果

9.2.6　单击按钮 button

单击按钮中<input>元素的 type 属性值为 button，用来激活相应的行为，或者为脚本和程序提供相应的值。语法结构如下所示。

```
<input type=" button"  属性="属性值" />
```

下面是一个使用单击按钮的实例。其代码如下所示。

例程 9-10　input-button.html

```
<body>
<h3>账号注册</h3>
用户名
<input type="text" size="25" name="name1" />
<input type="button" name="check" value="验证用户名是否可用" /><br />
  登录密码：<input type="password" name="username" size=30><br>
  确认密码：<input type="password" name="age" size=30 ><br>
<input type="submit" name="name2" value="提交" />
<input type="reset" name="name3" value="复位" />
</form>
</body>
```

其运行后的显示效果如图 9-12 所示。

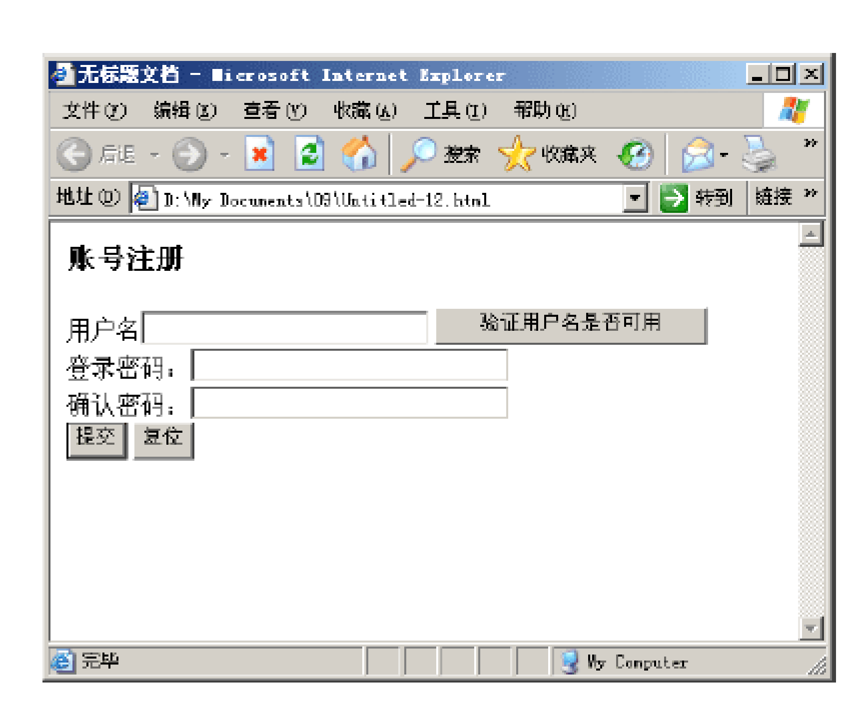


图 9-12　单击按钮的显示效果

9.2.7　复选框 checkbox

复选框中<input>元素的 type 属性值为 checkbox，用来提供用户多项选择的功能。语法结构如下所示。

```
<input type=" checkbox"  属性="属性值" />
```

下面是一个使用复选框的实例。其代码如下所示。

例程 9-11 input-checkbox.html

```
<body>
<h3>复选框　checkbox</h3>
<form action="edit.asp">
  选择你喜欢的书！<br />
  <input type="checkbox" checked="checked" name="chack" value="1" />武侠小说<br />
  <input type="checkbox" name="chack" value="2" />玄幻小说<br />
  <input type="checkbox" name="chack" value="3" />历史故事<br />
  <input type="checkbox" name="chack" value="4" />名人传记<br />
  <input type="checkbox" name="chack" value="5" />漫画<br />
  <input type="checkbox" name="chack" value="6" />杂志<br /><br />
  <input type="submit" name="name3" value="提交" />
  <input type="reset" name="name4" value="复位" />
</form>
</body>
```

其运行后的显示效果如图 9-13 所示。



图 9-13　复选框的显示效果

9.2.8　单选按钮 radio

单选按钮中<input>元素的 type 属性值为 radio，用来提供用户单项选择的功能。语法结构如下所示。

```
<input type="radio"  属性="属性值" />
```

用户只能够选择一个按钮，如果点选其他按钮，则原来选择会自动取消。下面是一个使用单选按钮的实例。其代码如下所示。

例程 9-12 input-radio.html

```
<body>
<h2>心理小测试：拿钥匙的方式</h2>
 <hr>
 下班回家,家门的钥匙怎么处理,可以反映出你的性格的中心部分呢。试试看吧：
 <hr>
 <form name="example" action="deal.asp" method="post">
<input type="radio" name="test" value="answerA" checked>下班前就把钥匙放在口袋里<br>
<input type="radio" name="test" value="answerB" >快到家门口的时候找钥匙<br>
<input type="radio" name="test" value="answerC" >到了门口再找钥匙<br>
<input type="radio" name="test" value="answerD" >从来不想这个问题<br>
<input type="submit" name="name3" value="提交" />
<input type="reset" name="name4" value="复位" />
</form>
</body>
```

其运行后的显示效果如图 9-14 所示。

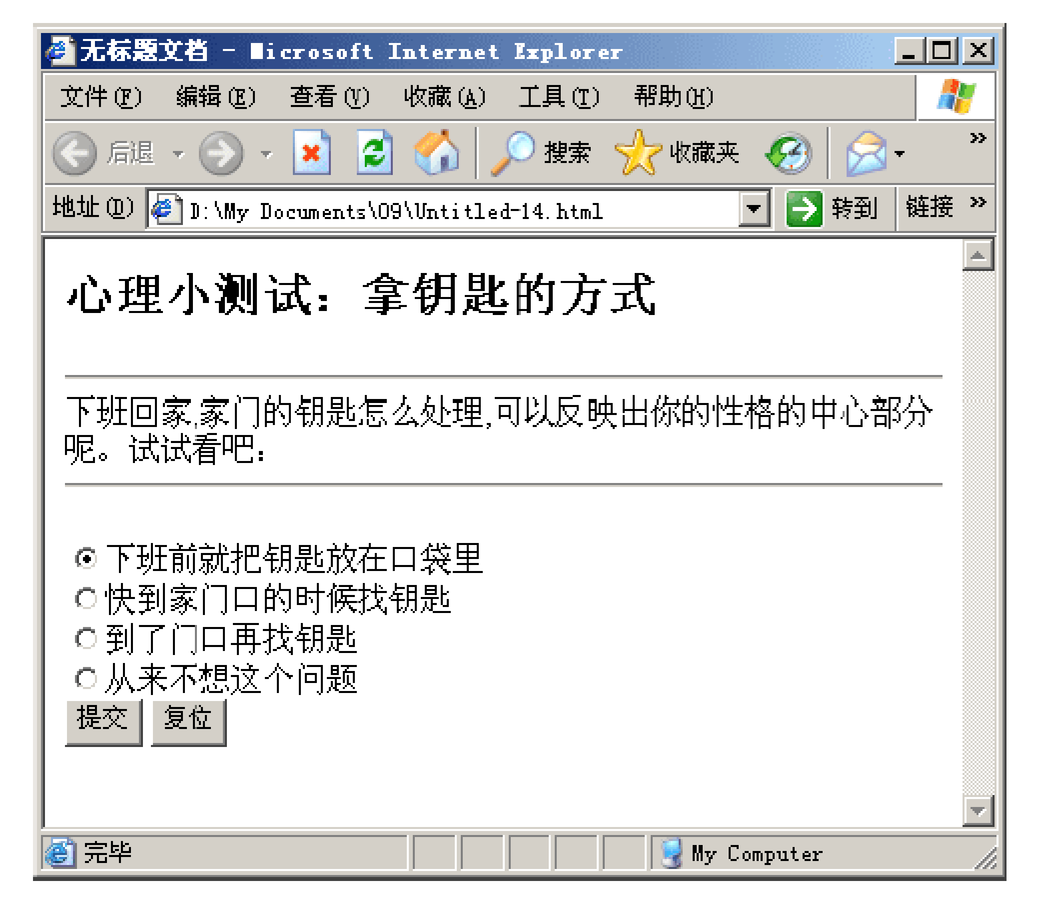


图 9-14　单选按钮的显示效果

9.2.9　隐藏区域 hidden

隐藏区域中<input>元素的 type 属性值为 hidden，使表单的内容不出现在页面之中，但是依然可以传递参数。语法结构如下所示。

```
<input type="hidden"  属性="属性值" />
```

下面是一个使用隐藏区域的实例。其代码如下所示。

例程 9-13　input-hidden.html

```
<body>
 下面是几种不同属性的文字字段:
<form name="example" action="deal.asp" method="post">
     姓名:<input type="text" name="username" size=35>
     <br>
     年龄:<input type="text" name="age" size=35 maxlength=2>
     <br
     个人主页:<input type="text" name="privateweb" size=15 maxlength=30 value="http://">
     <!--添加隐藏内容-->
     <input type="hidden" name="page_id" value="example ">
    <br >
<input type="submit" name="name3" value="提交" />
<input type="reset" name="name4" value="复位" />
</form>
</body>
```



注意

隐藏区域是不占有空间的。

其运行后的显示效果如图 9-15 所示。

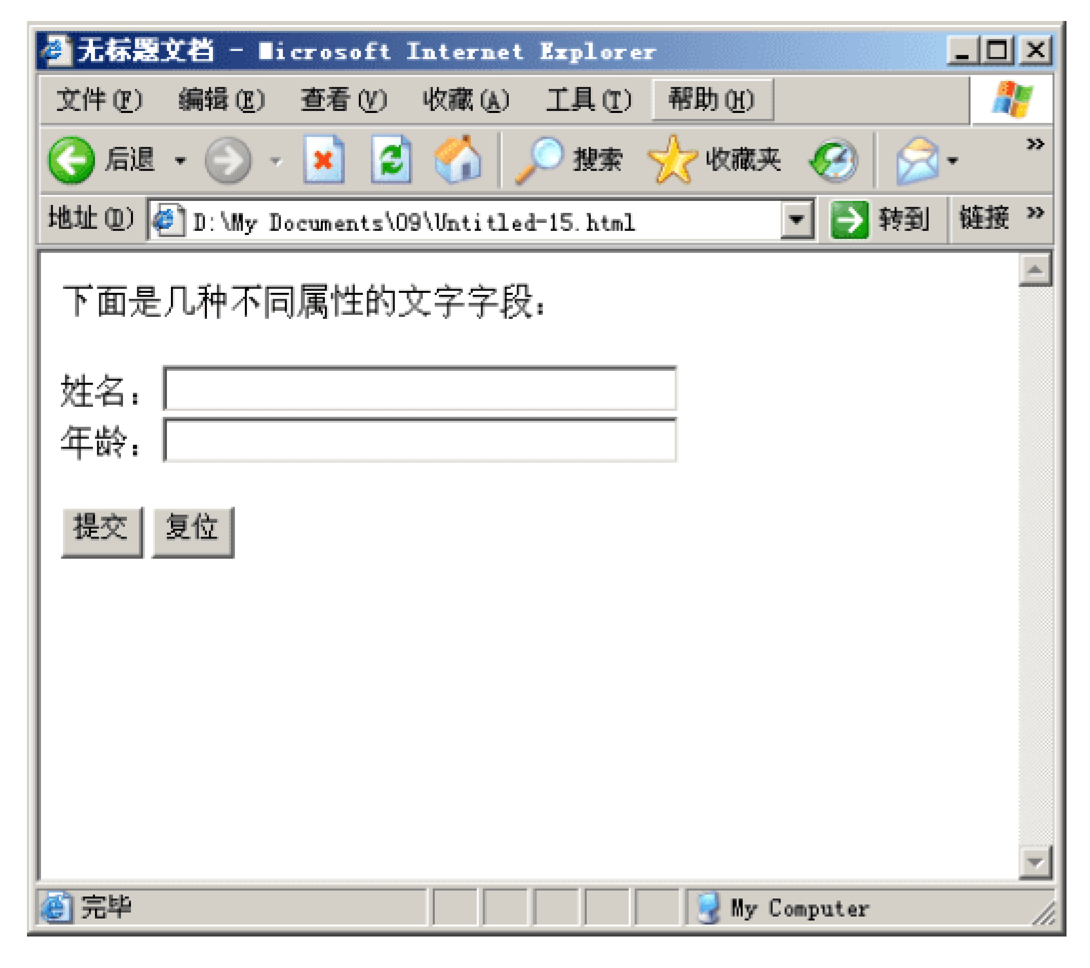


图 9-15 隐藏区域的显示效果

9.3 <input>元素的属性

在 9.2 节中，主要讲解了<input>元素的 type 属性。接下来讲解下<input>元素中其他部分属性。

9.3.1 只读属性 readonly

只读属性 readonly 用来指定<input>元素中的内容为只读。使用 readonly 属性后，用户将不能更改表单中的相应内容。但是表单中的内容仍然可以传递。语法结构如下所示。

```
<input readonly="readonly"  属性="属性值" />
```

下面是一个使用只读属性的实例。其代码如下所示。

例程 9-14 input-readonly.html

```
<body>
<h3>一个表单的示例</h3>
<form action="edit.asp">
 您获得的用户名：
<input type="text" size="19" name="name1" value="abcde" readonly="readonly" /><br />
 请添加您的密码：
<input type="password" size="20" name="name2" /><br /><br />
<input type="submit" name="name3" value="提交" />
<input type="reset" name="name4" value="复位" />
</form>
</body>
```

其运行后的显示效果如图 9-16 所示。

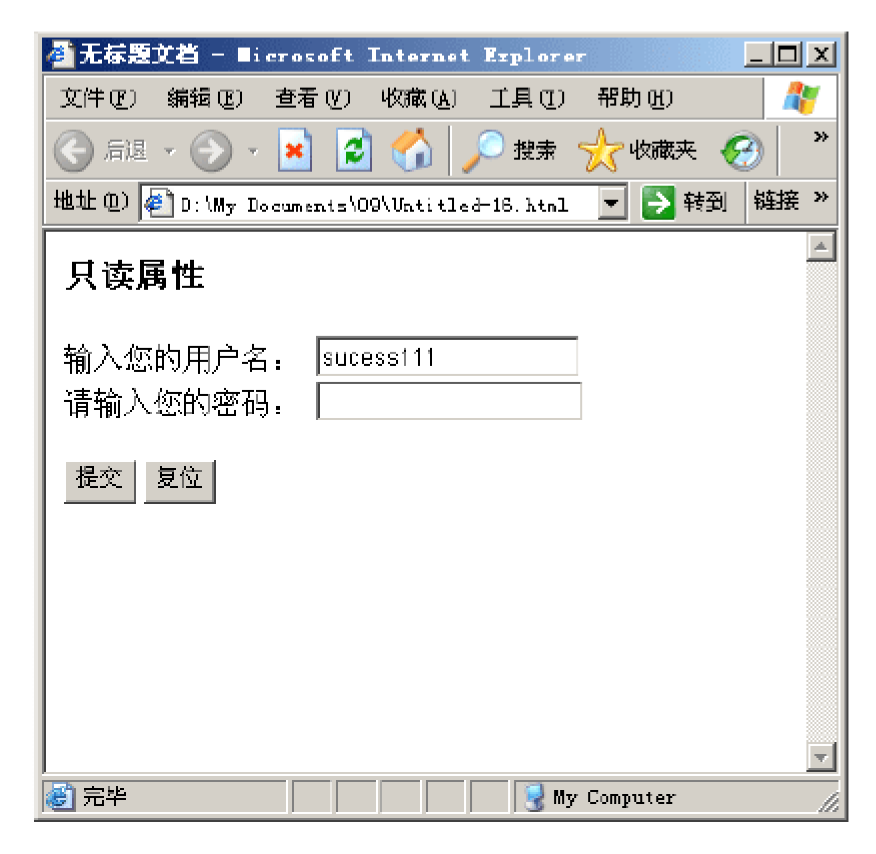


图 9-16 使用 readonly 属性的显示效果

9.3.2 不可用属性 disabled

不可用属性 disabled 用来指定<input>元素中的内容为不可用。使用 disabled 属性后，用户将不能更改表单中的相应内容。同时表单中的内容将不能被传递。语法结构如下所示。

```
<input disabled =" disabled"  属性="属性值" />
```

下面是一个使用不可用属性的实例。其代码如下所示。

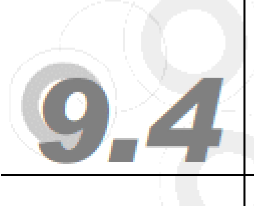
例程 9-15 input-disabled.html

```
<body>
<h3>不可用属性</h3>
<form action="edit.asp">
  您的通行证：
<input type="text" size="20"   value="abcde" disabled="disabled" /><br />
  输入您的密码：
<input type="password" size="22" /><br /><br />
<input type="submit" name="name3" value="提交" disabled="disabled" />
<input type="reset" name="name4" value="复位" />
</form>
</body>
```

其运行后的显示效果如图 9-17 所示。



图 9-17 使用 disabled 属性的显示效果



9.4 选择列表条目元素<option>

选择列表条目元素<option>用来定义下拉列表中的项目。语法结构如下所示。

```
<select 属性="属性值'>
  <option 属性="属性值'>选择列表内容</option>
</select>
```



注意

<option>元素必须要在<select>元素中使用。

下面是一个使用<option>元素的实例。其代码如下所示。

例程 9-16 input-disabled.html

```
<body>
 这里是 126 邮箱的注册页面:
<form action="mailto:abcd@163.com" name="research" method="post">
<p>用户名:
     <input name="username" type="text" size=20>
</p>
<p>登录密码:
     <input name="password" type="password" size=20>
</p>
<p>
   重复密码:
     <input name="password2" type="password" size=20>
<p>为了找回密码，需要你输入下面证件类型:
<select name="cardtype">
      <option value="id_card" selected>身份证
      <option value="stu_card">学生证
      <option value="drive_card">驾驶证
      <option value="other_card">其他证件
</select>
</p>
<p>证件号码:
     <input name="cardnum" type="text" size=20 maxlength=35>
</p>
<p>联系方式:
     <input name="touch" type="text" size=20 maxlength=50>
</p>
<p>关心的栏目:
<select name="content" size=5 multiple>
      <option value="M1" selected>体育栏目
      <option value="M2">科技内容
```

```
        <option value="M3">新闻栏目
        <option value="M4">卡通动漫
        <option value="M5" selected>财经证券
        <option value="M6">娱乐八卦
        <option value="M7">金融投资
        <option value="M8">旅游交友
</select>
<p>
        <input type="submit" name="Submit" value="提交">
                   
        <input type="reset" name="reset" value="重置">
</p>
</form>
</body>
```

其运行后的显示效果如图 9-18 所示。

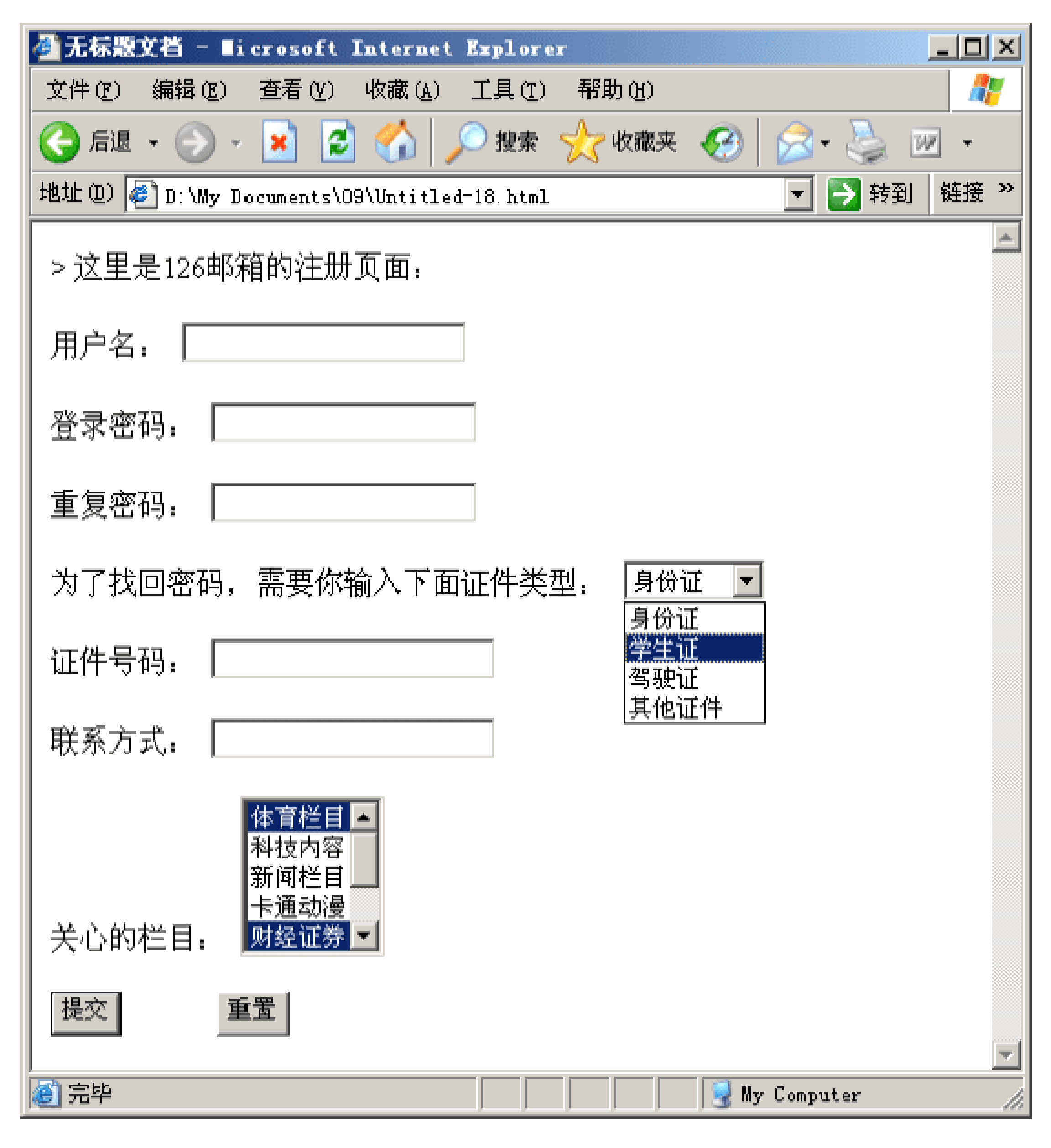


图 9-18　使用<option>元素的显示效果

9.5 按钮元素<button>

按钮元素<button>用来定义一个按钮。在<button>元素中，不但可以使用文本内容，也可以使用图片等其他元素。语法结构如下所示。

```
<button>按钮内容</button>
```

下面是一个使用<button>元素的实例。其代码如下所示。

例程 9-17 button.html

```
<body>
<h3>对本页面的留言</h3>
<form action="edit.asp">
  用户名：<input type="text" size="20" name="name1" /><br />
  用户留言：<textarea cols="40" rows="10" name="name2"></textarea><br /><br />
<button name="name3" type="submit">按扭</button>
</form>
</body>
```

其运行后的显示效果如图 9-19 所示。



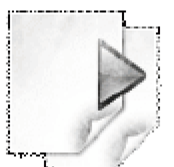
图 9-19 使用<button>元素的显示效果

上面用到了按钮元素<button>的类型属性 type 来定义按钮元素的类型。

9.6 选择列表元素<select>

选择列表元素<select>用来在页面中定义一个可以选择的下拉列表。语法结构如下所示。

```
<select 属性="属性值'>
  <option 属性="属性值'>选择列表内容</option>
</select>
```



注意

<select>元素必须和<option>元素一起使用。

下面是一个使用<select>元素的实例。其代码如下所示。

例程 9-18　select.html

```
<body>
<h3>注册论坛会员</h3>
<form action="edit.asp">
 您的账号：<input type="text" size="30"   /><br />
 输入密码：<input type="password" size="31"   /><br /><br />
<p>为了找回密码，需要你输入下面证件类型：
<select name="cardtype">
 <option value="id_card" selected>身份证
 <option value="stu_card">学生证
 <option value="drive_card">驾驶证
 <option value="other_card">其他证件
</select>
</p>
证件号码：<input type="text" size="30"   /><br />
<input type="submit" name="name3" value="提交" />
<input type="reset" name="name4" value="复位" />
</form>
</body>
```

其运行后的显示效果如图 9-20 所示。

当单击<select>元素中带有向下的箭头图标时，效果如图 9-21 所示。

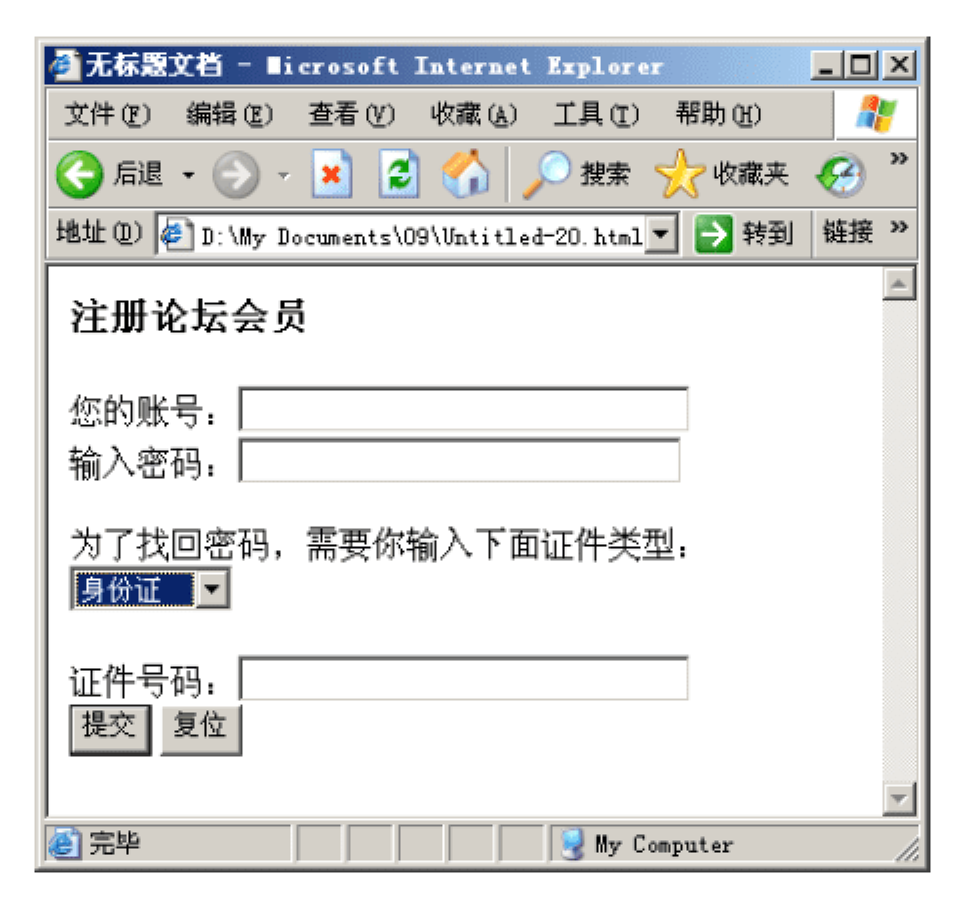


图 9-20　使用<select>元素的显示效果

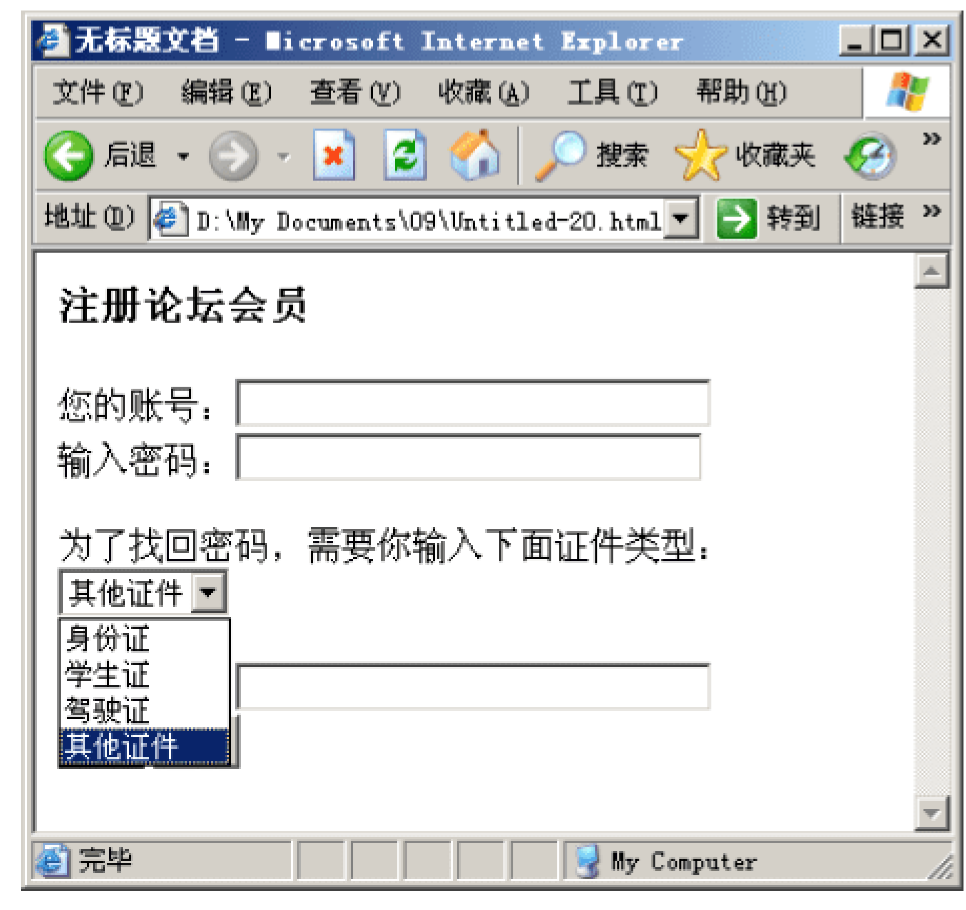


图 9-21　展开列表后的显示效果

9.6.1　高度属性 size

高度属性 size 用来定义<select>元素中显示下拉列表区域的高度。在本章前面例程 3-20 讲解的列表元素中，size 属性一直用来定义显示区域的宽度，所以要特别注意其使用环境，防止混淆。语法结构如下所示。

```
<select size="数字值'>
  <option 属性="属性值'>选择列表内容</option>
</select>
```

数字值代表的含义是：文本行的数目。下面是一个使用 size 属性的实例。其代码如下所示。

例程 9-19 select-size.html

```
<body>
<h3>注册论坛会员</h3>
<form action="edit.asp">
  您的账号：<input type="text" size="30"   /><br />
  输入密码：<input type="password" size="31"   /><br /><br />
<p>为了找回密码，需要你输入下面证件类型：
<select name="name1" size="5">
 <option value="id_card" selected>身份证
 <option value="stu_card">学生证
 <option value="drive_card">驾驶证
 <option value="other_card">其他证件
</select>
</p>
   证件号码：<input type="text" size="30"   /><br />
  <input type="submit" name="name3" value="提交" />
  <input type="reset" name="name4" value="复位" />
</form>
</body>
```

其运行后的显示效果如图 9-22 所示。

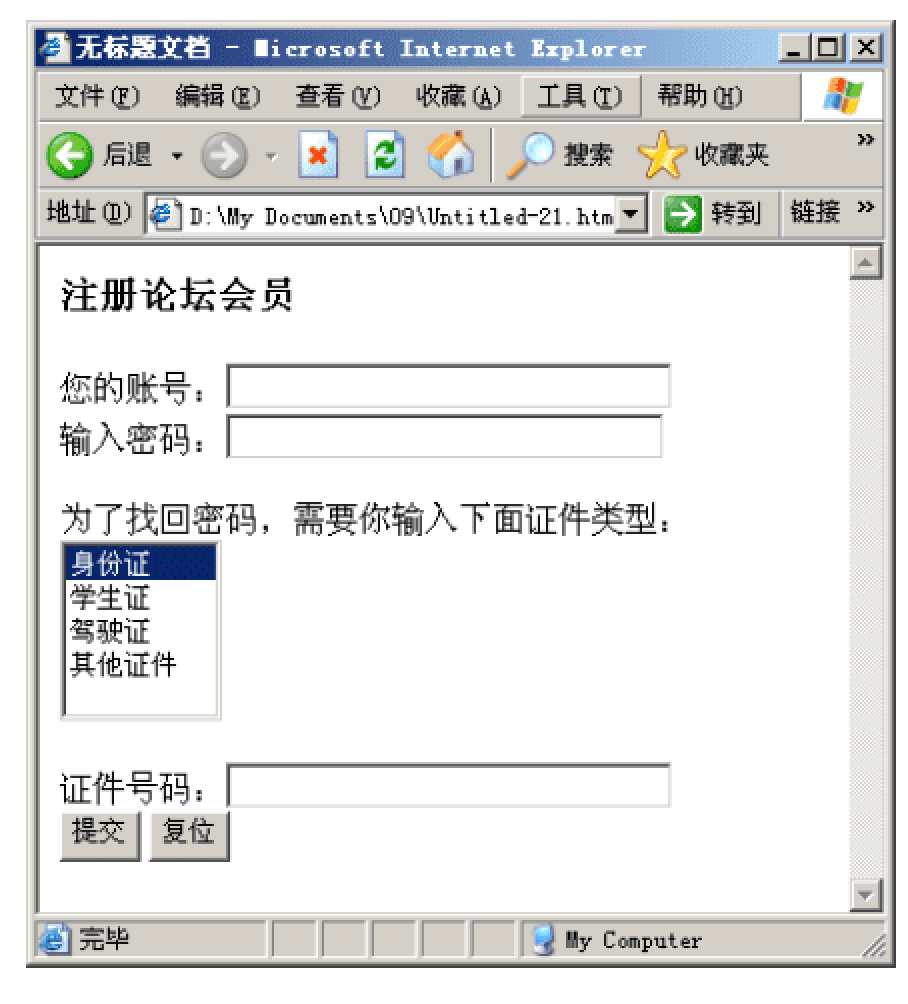


图 9-22　使用 size 属性的显示效果

9.6.2　多项选择属性 multiple

列表项的设置方法与下拉菜单类似，不同的是列表项在页面中可以显示出几条信息，一旦超出这个信息数量，在列表右侧会出现滚动条，拖动滚动条能看到所有的选项。语法结构如下；

```
    <select name="列表项名称" size="显示的列表项数" multiple>
        <option value="选项值" selected>选项显示内容
        <option value="选项值">选项显示内容
        ……
    </select>
```

这个属性的使用请参考实例 9-16，这里不再做实例演示。

说明

在该语法中，size 设定页面中的最多列表项数，当超过这个值时会出现滚动条。multiple 表示这一列表可以进行多项选择。选项值是提交表单时的值，而选项显示内容才是真正在页面中显示的选项。

9.7 文本区域元素<textarea>

除了前面讲解的两大类控件外，还有一种特殊定义的文本样式，称为文字域或文本域。它与文字字段的区别在于可以添加多行的文字，从而可以输入更多的文本。这类控件在一些留言板中最为常见。

语法结构如下：

```
<textarea name="文本域名称" value="文本域默认值" rows=行数  cols=列数>
</textarea>
```

说明

在该语法中，rows 是指文本域的行数，也就是高度值，当文本内容超出这一范围会出现滚动条；cols 设置文本域的列数，也就是其宽度。

9.7.1 宽度属性 cols

宽度属性 cols 用来定义文本区域的宽度。语法结构如下所示。

```
<textarea cols="数字值">元素内容</textarea>
```

其中数字值的含义是字符数目。下面是一个使用 cols 属性的实例。其代码如下所示。

例程 9-20 textarea-cols.html

```
<body>
<h3>文本区域元素</h3>
<form action="edit.asp">
 用户名：<input type="text" size="30" name="name1" /><br />
 留言：<textarea cols="35" rows="10" name="name2"></textarea><br />
 <input name="name3" type="submit" value="提交" />
 <input name="name4" type="reset" value="重置" />
</form>
</body>
```

其运行后，在文本区域中输入文本时的显示效果如图 9-23 所示。

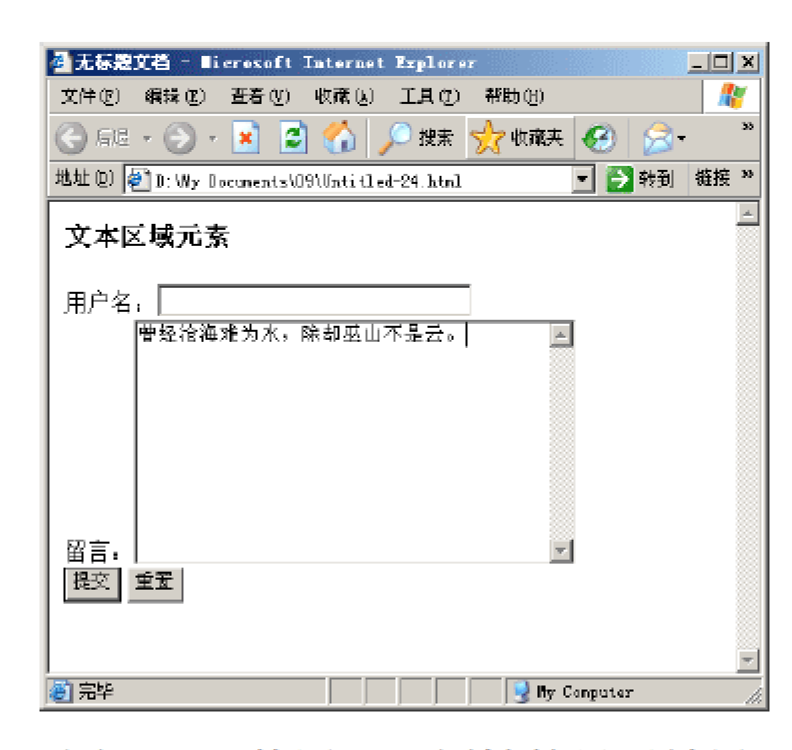


图 9-23 使用 cols 属性的显示效果

9.7.2 高度属性 rows

高度属性 rows 用来定义文本区域的高度。语法结构如下所示。

```
<textarea rows="数字值">元素内容</textarea>
```

其中数字值的含义是行的数目。下面是一个使用 rows 属性的实例。其代码如下所示。

例程 9-21 textarea-rows.html

```
<body>
<h3>一个表单的示例</h3>
<form action="edit.asp">
留言：<textarea cols="40" rows="10" name="name2"></textarea>
</form>
</body>
```

其运行后，在文本区域中输入文本时的显示效果如图 9-24 所示。

从图 9-24 中可以看出 rows 属性取值的含义，此时取值为 10，所以刚好可以显示 10 行。含有更多的内容时将显示滚动条，如图 9-25 所示。

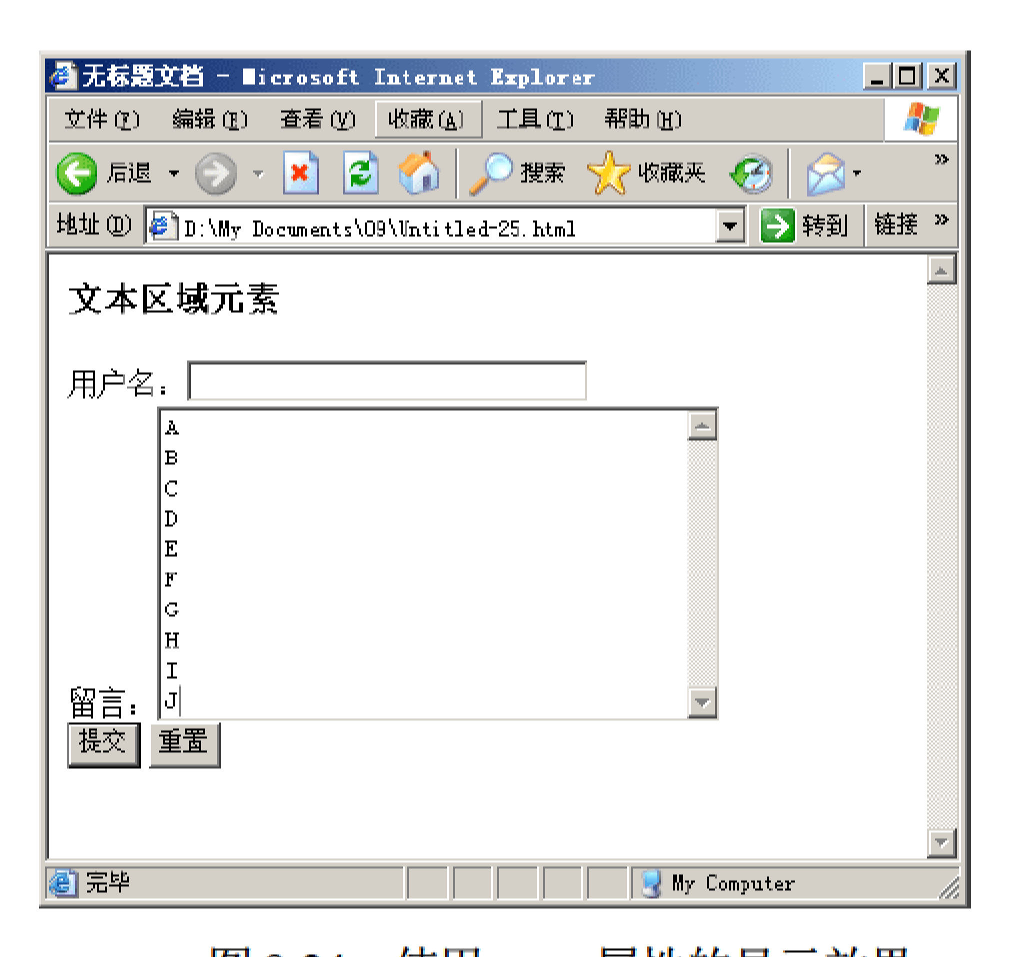


图 9-24 使用 rows 属性的显示效果

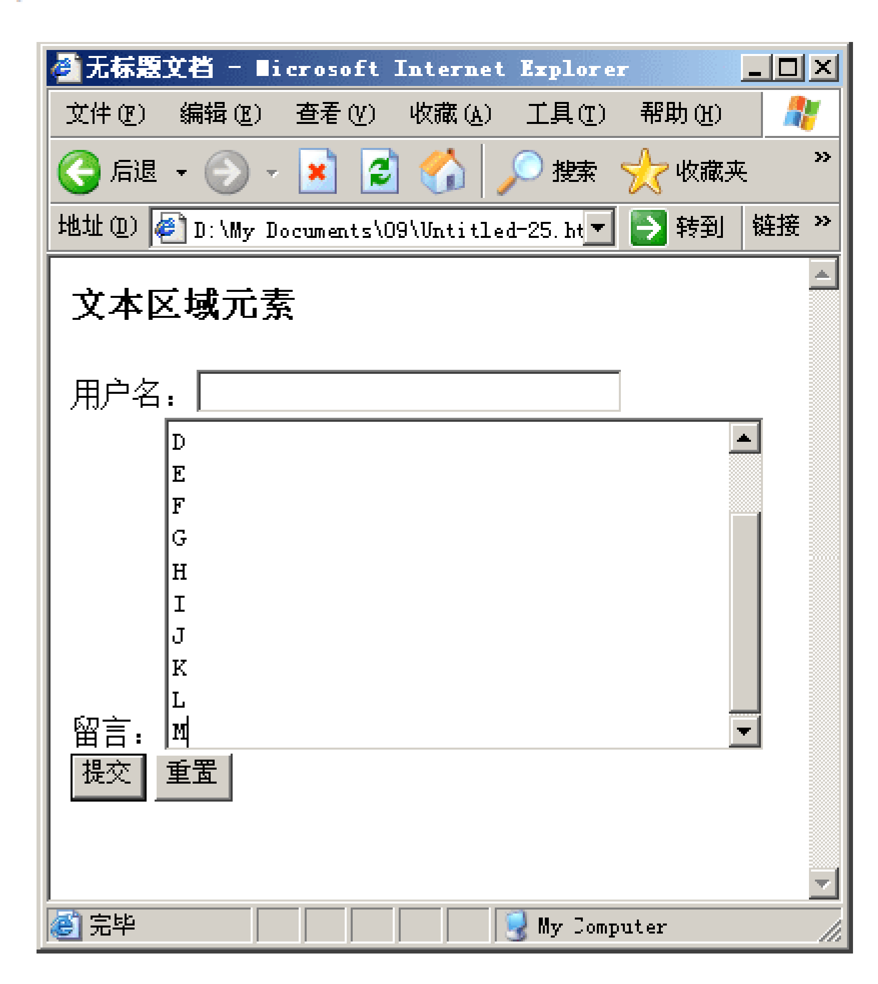


图 9-25 内容超出高度的显示效果

9.8 表单标记元素<label>

表单标记元素<label>用来定义一个关联的标记，将信息附属在表单控件上。例如，定义关联表单的文本，当单击文本时，实现表单的选择。或者定义表单的热键等。语法结构如下所示。

```
<label>元素内容</label>
```

其中部分属性的含义和用法如下所示。

9.8.1 定义目标属性 for

定义目标属性 for 用来关联文档中相应的表单。使用 for 属性时，目标表单要有唯一的 id 标识。所以通过 for 属性可以将<label>元素与唯一一个表单关联，但是一个表单却可以关联几

个<label>元素。语法结构如下所示。

```
<label for="目标表单的 id">元素内容</label>
```

下面是一个使用 for 属性的实例。其代码如下所示。

例程 9-22　label-for.html

```
<body>
<h3>定义目标属性</h3>
<form id="form1" name="form1" method="post" action="">
 选择你喜欢的动物<br />
<input type="checkbox" name="check" value="1" id="check1" />
<label for="check1">老虎</label><br />
<input type="checkbox" name="check" value="2" id="check2" />
<label for="check2">狮子</label><br />
<input type="checkbox" name="check" value="3" id="check3" />
<label for="check3">鹦鹉</label><br />
<input type="checkbox" name="check" value="4" id="check4" />
<label for="check4">狗</label><br />
<input type="checkbox" name="check" value="5" id="check5" />
<label for="check5">猫</label><br /><br />
<input type="submit" name="name" value="提交" />
</form>
</body>
```

其运行后的显示效果如图 9-26 所示。

从图 9-26 中可以看出，for 属性对<label>元素的显示效果并没有影响。但是由于使用 for 属性实现了表单和<label>元素的关联，所以当单击文本内容时，相应的复选框就会被选中。

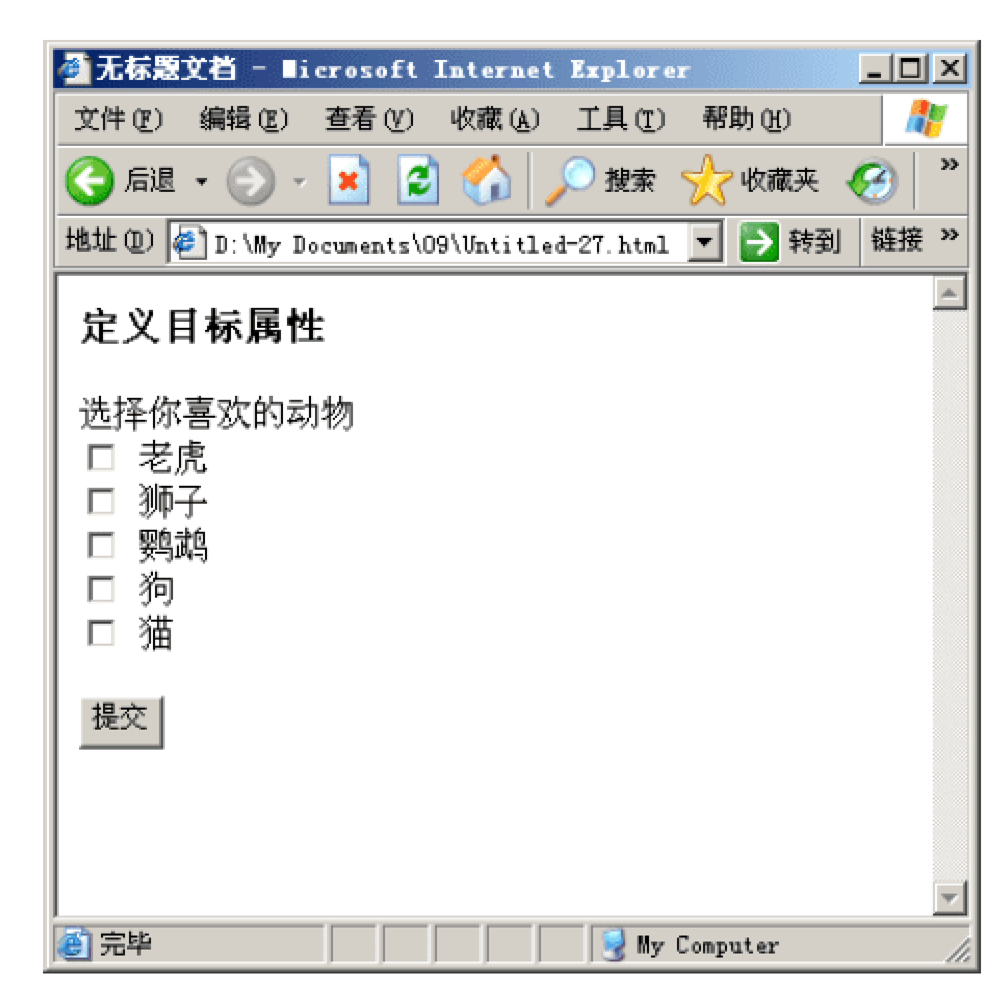


图 9-26　使用 for 属性的显示效果

9.8.2　定义热键属性 accesskey

定义热键属性 accesskey 用来定义激活关联表单的热键。定义热键之后，只需使用键盘上的 Alt 键加热键就可以选择相应的表单。语法结构如下所示。

```
<label for="目标表单的id" accesskey="键的名称">元素内容</label>
```

下面是一个使用 accesskey 属性的实例。其代码如下所示。

例程 9-23　label-accesskey.html

```
<body>
<h3>定义热键属性</h3>
<form id="form1" name="form1" method="post" action="">
```

```
选择你喜欢的动物<br />
<input type="checkbox" name="check" value="1" id="check1" />
<label for="check1" accesskey="z">老虎</label><br />
<input type="checkbox" name="check" value="2" id="check2" />
<label for="check2" accesskey="b">狮子</label><br />
<input type="checkbox" name="check" value="3" id="check3" />
<label for="check3" accesskey="c">狼</label><br />
<input type="checkbox" name="check" value="4" id="check4" />
<label for="check4" accesskey="d">狗</label><br />
<input type="checkbox" name="check" value="5" id="check5" />
<label for="check5" accesskey="e">猫</label><br /><br />
<input type="submit" name="name" value="提交" />
</form>
</body>
```

其运行后，同时按下 Alt 键和 A 键时的显示效果如图 9-27 所示。

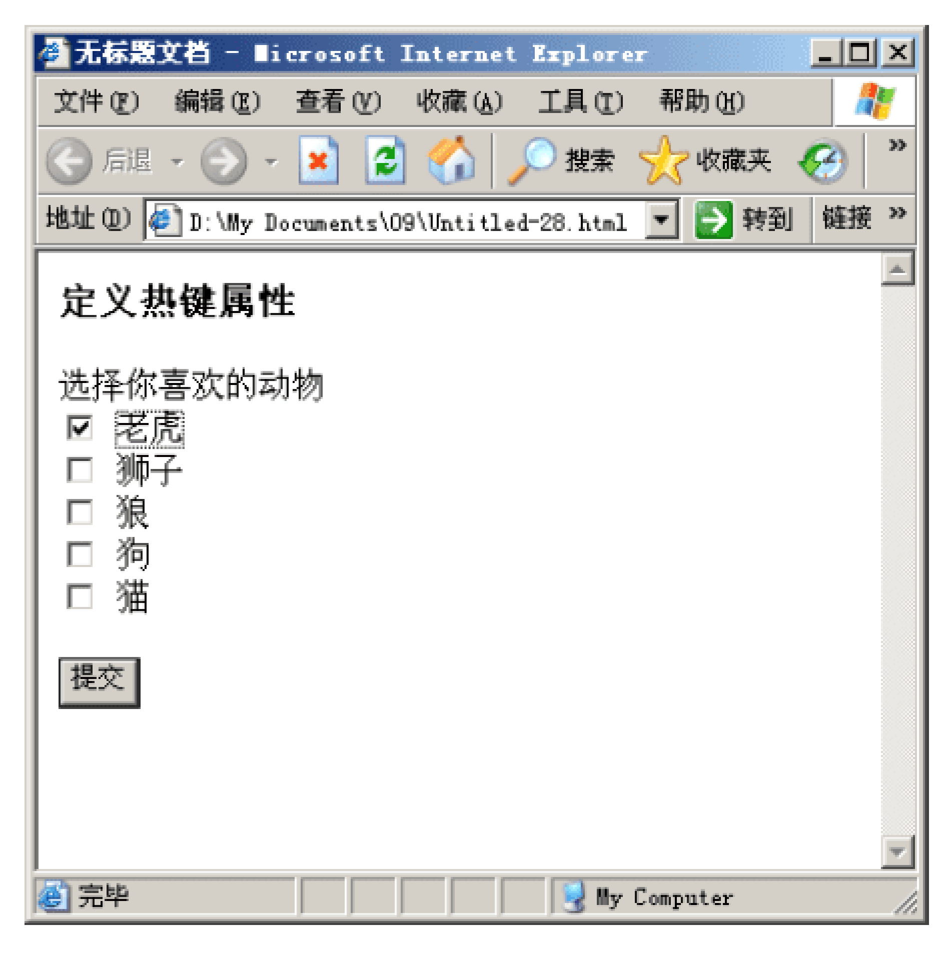


图 9-27 使用 accesskey 属性的显示效果

使用热键激活表单的方法，在中文页面中会存在局限。因为没有方便的方法，能够指示出表单使用的热键。所以用户将很难操作。

9.9 本章习题

一、选择题

1．如果要在表单里创建一个普通文本框，以下写法中正确的是(　　)。

A. < Input >

B. < Input type="password">

C. < Input type="checkbox">

D. < Input type="radio">

2．以下有关表单的说明中，错误的是(　　)。

A. 表单通常用于搜集用户信息。

B. 在 form 标记符中使用 action 属性指定表单处理程序的位置。

C. 表单中只能包含表单控件，而不能包含其他诸如图片之类的内容。

D. 在 form 标记符中使用 method 属性指定提交表单数据的方法。

3．在指定单选框时，只有将以下_____属性的值指定为相同，才能使它们成为一组。

A. type　B. name　C. value　D. checked

4．创建选项菜单应使用以下标记符(　　)。

A. select 和 option

B. Input 和 label

C. Input

D. Input 和 option

5．以下有关按钮的说法中，错误的是(　　)。

A. 可以用图像作为提交按钮。

B. 可以用图像作为重置按钮。

C. 可以控制提交按钮上的显示文字。

D. 可以控制重置按钮上的显示文字。

6．若要产生一个 4 行 30 列的多行文本域，以下方法中，正确的是(　　)。

A. <Input type="text" Rows="4" Cols="30" Name="txtintrol">

B. <TextArea Rows="4" Cols="30" Name="txtintro">

C. <TextArea Rows="4" Cols="30" Name="txtintro"></TextArea>

D. <TextArea Rows="30" Cols="4" Name="　　txtintro"></TextArea>

7．以下表单控件中，不是由 INPUT 标记符创建的为(　　)。

A. 单选框

B. 口令框

C. 选项菜单

D. 提交按钮

8．要给表单控件设置标签，以下代码中正确的是(　　)。

A. <INPUT type="checkbox" name="news"><LABEL for="news">新闻</LABEL>

B. <INPUT type="checkbox" for="news"><LABEL id="news">新闻</LABEL>

C. <INPUT type="checkbox" for="news"><LABEL name="news">新闻</LABEL>

D. <INPUT type="checkbox" id="news"><LABEL for="news">新闻</LABEL>

二、填空题

1．表单对象的名称由_________属性设定；提交方法由_________属性指定；若要提交大数据量的数据，则应采用_________方法；表单提交后的数据处理程序由_________属性指定。

2．表单是 Web___________和 Web___________之间实现信息交流和传递的桥梁。

3．表单实际上包含两个重要组成部分：一是描述表单信息的________，二是用于处理表单数据的服务器端__________。

4．表单使_________可以与_______进行交互，是收集客户信息和进行网络调查的主要途径。

三、实战练习

1. 已知页面效果如图 9-28 所示，请填写以下 HTML 代码中留下的空白。



图 9-28　页面效果

```
    <form>
    <table border bgcolor="#eeeeee">
<tr><td  ____   align=center><b>情况登记</b>
<tr><td>姓名：<td> ____________
<tr><td>密码：<td><INPUT size="20" _________value="pwd">
<tr><td>性别：<td><INPUT name="sex" type=radio>男 <INPUT name="sex" type=radio>女
<tr><td> <td><INPUT ________value="提交"> <INPUT type="reset" ______>
    </table>
    </form>
```

2. 已知页面效果如图 9-29 所示(其中的细线效果为 1 像素粗细，颜色为黑色)，请填写以下 HTML 代码中留下的空白。



图 9-29　页面效果

```
    <FORM>
    <TABLE __________  cellpadding="5" align="center" bgcolor="#dddddd">
    <TR>
      <TD  _________   colspan="2"><B>个人信息</B>
    <TR bgcolor="white">
<TD>姓名：<TD><INPUT size="20">
```

```
<TR bgcolor="white"><TD>密码：<TD><INPUT size="20"  ________  value="pwd">
<TR bgcolor="white">
<TD>性别：<TD><INPUT name="sex" type=radio>男 <INPUT  ______  type=radio>女
<TR><TD> <TD><INPUT value="提交"  ________  > <INPUT type="reset" value="
重写">
</TABLE>
</FORM>
```

Chapter 10

框架元素

所谓框架就是把页面分为几个部分，各个部分之间是相互独立的页面，却又互相有关联。用户在浏览这种页面的时候，当对其中某一个部分进行操作，如浏览、下载的时候，其他页面会保持不变，这样的页面就被称为框架结构的页面，也称为多窗口页面。

框架元素的最主要功用是用来“分割”页面窗口，使每个“小窗口”能显示不同的 HTML 文件，这样的页面结构就称为框架结构的页面，而这些“小窗口”就被称为框架的“窗口”。在 HTML 中，框架元素包括<frameset>、<frame>、<iframe>等。

本章主要内容有:

◎ 各种框架元素及其属性的使用方法和表现效果。
◎ 能够熟练分隔页面窗口。
◎ 学会框架的嵌套。
◎ 学会框架中的复杂链接。

10.1 框架集元素<frameset>

框架集元素<frameset>用来在页面中定义一个框架集合。同时使用<frameset>元素也可以控制框架窗口之间的距离，窗口的大小等。语法结构如下所示。

```
<html>
    <head>
    <title>……</title>
    </head>
    <frameset>
        <frame>
        <frame>
        ……
    </frameset>
    </html>
</html>
```

从上面的语法结构中可以看到，在使用框架的页面中，<body>主体标记被框架标记<frameset>所代替。而对于框架页面中包含的每一个框架，都是通过<frame>标记来定义的。

下面是一个使用<frameset>元素的实例。其代码如下所示。

例程 10-1 frameset.html

```
01  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
02  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
03  <html xmlns="http://www.w3.org/1999/xhtml">
04  <head>
05  <meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
06  <title>框架集元素</title>
07  </head>
08  <frameset rows="*,80" cols="*" frameborder="no" border="0" framespacing="0">
09  <frameset cols="80,*" frameborder="no" border="0" framespacing="0">
10    <frame src="left.html" name="leftframe" scrolling="no" noresize="noresize" id="leftframe"
title="leftframe" />
11  <frame src="main.html" name="mainframe" id="mainframe" title="mainframe" />
12  </frameset>
13 <frame src="bottom.html" name="bottomframe" scrolling="no" noresize="noresize" id="bottomframe"
title="bottomframe" />
14  </frameset>
15  </html>
```

该实例中，08 到 14 行中使用了嵌套的<frameset>元素，定义了一个含有 3 个独立区域的框架页面。其运行后的显示效果，如图 10-1 所示。

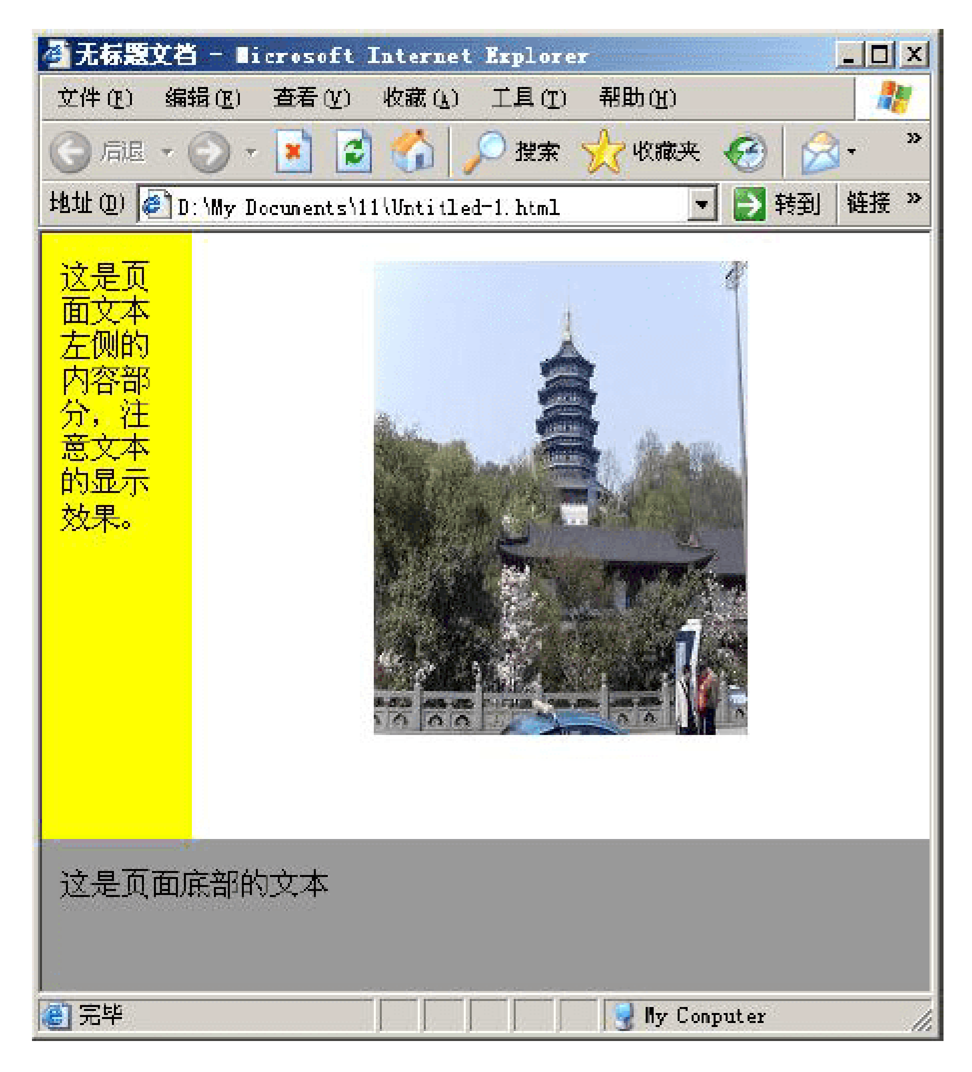


图 10-1　使用<frameset>元素的显示效果

10.1.1　行属性 rows

框架集元素的行属性 rows 用来指定框架集中行的数目。在框架集中，必须使用 rows 属性和列属性 cols 其中的一个(关于列属性 cols 的内容，将在 10.1.2 小节中讲解)。语法结构如下所示。

```
<frameset rows="高度值 1，高度值 2,…'>其他框架元素与框架集</frameset>
```

其中 rows 属性的取值，是用英文“,”分隔的一组数值(包括百分比值和通配符“*”)。其中每个数值代表一个框架区域的高度。

注意

可能会存在数字值高度之和大于或者小于浏览器窗口的情况。此时浏览器不会自动调整窗口大小，或者增加滚条来使用框架，而是会按照每个框架区域所占有的高度比例来显示框架内容，具体的显示效果，将在实例中演示。

下面是一个使用 rows 属性的实例。其代码如下所示。

例程 10-2　frameset-rows.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>水平分割窗口的效果</title>
</head>
<frameset rows="30%,70%">
 <frame>
 <frame>
```

```
09  </frameset>
10  </html>
```

该实例中，08 行指定了页面中 2 个框架区域的高度分别是 30%、70%。其代码运行后，显示效果，如图 10-2 所示。

如果将 rows 属性的取值改为总的高度大于 100%，这时框架页面的显示高度并不会变化，只是每个框架区域的高度会重新分配。

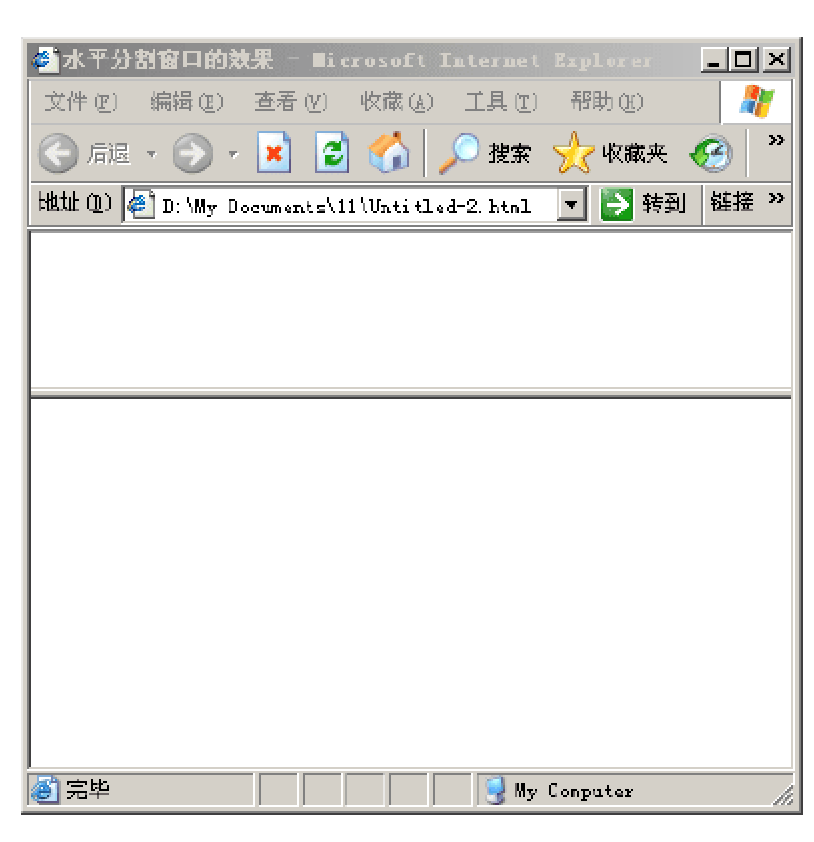


图 10-2　使用百分比值的显示效果

10.1.2　列属性 cols

框架集元素的列属性 cols 用来指定框架集中列的数目。语法结构如下所示。

```
<frameset cols="框架窗口的宽度,框架窗口的宽度,……">
        <frame>
        <frame>
        ……
    </frameset>
```

其中 cols 属性的取值，是用英文“,”分隔的一组数值(包括百分比值和通配符“*”)。其中每个数值代表一个框架区域的宽度。

注意

可能会存在数字值的宽度之和大于或者小于浏览器窗口的情况。此时浏览器不会自动调整窗口大小，或者增加滚条来使用框架，而是会按照每个框架区域所占有的宽度比例来显示框架内容，具体的显示效果，将在实例中演示。

下面是一个使用 cols 属性的实例。其代码如下所示。

例程 10-3　frameset-cols.html

```
01  <frameset cols="20%,55%,25%">
     <frame>
     <frame>
     <frame>
02  </frameset>
03  </html>
```

该实例中，01 行指定了页面中 3 个框架区域的宽度分别是 20%、55%、25%。其代码运行后，显示效果如图 10-3 所示。

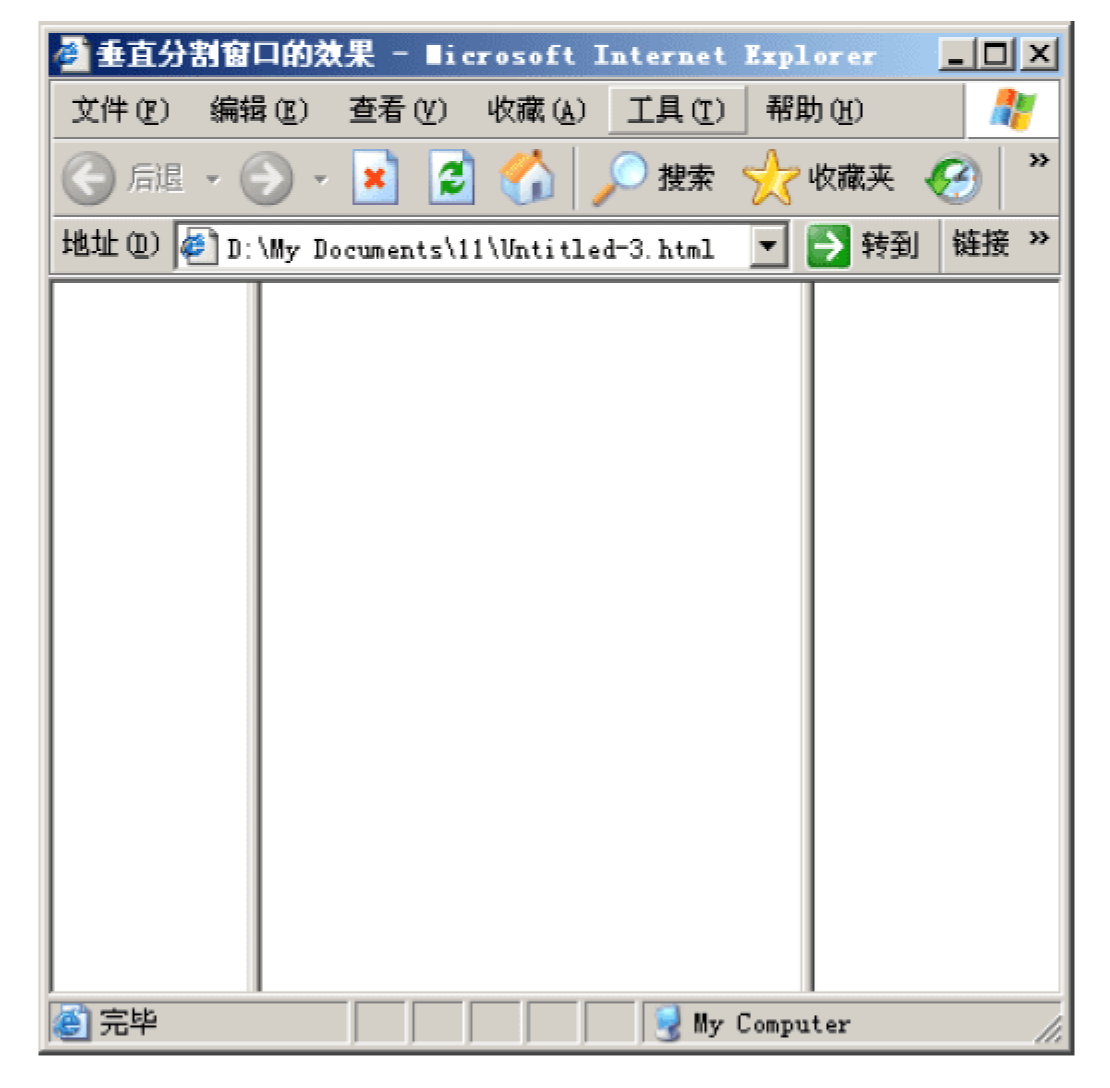


图 10-3　使用百分比值的显示效果

如果将 cols 属性的取值改为总的宽度大于 100%，这时框架页面的显示宽度并不会变化，只是每个框架区域的宽度会重新分配。大家可以自己体会一下。使用数字值的情况和使用百分比值的情况基本相同。

学习了前面的知识后，就可以在一个页面中设置既有水平分割的框架，又有垂直分割的框架。

语法结构如下：

```
<frameset rows="框架窗口的高度,框架窗口的高度,……">
        <frame>
          <frameset cols="框架窗口的宽度,框架窗口的宽度,……">
            <frame>
            <frame>
            ……
          </frameset>
        <frame>
        ……
</frameset>
```

当然，也可以先进行垂直分割，再进行水平分割。其语法如下：

```
<frameset cols="框架窗口的宽度,框架窗口的宽度,……">
        <frame>
          <frameset rows="框架窗口的高度,框架窗口的高度,……">
            <frame>
            <frame>
            ……
          </frameset>
        <frame>
        ……
</frameset>
```

在页面同时使用这两种分隔方法时，主要是需要注意窗口大小的设置与窗口个数的统一。实例代码如下。

例程 10-4 frameset-cols.html

```
<frameset rows="30%,70%">
 <frame>
   <frameset cols="20%,55%,25%">
      <frame>
      <frame>
      <frame>
   </frameset>
</frameset>
</html>
```

由代码中可以看出，首先将页面进行水平分割成上下两个窗口，而下面的框架又被垂直分割成 3 个窗口。因此下面的框架标记<frame>被框架集标记代替。运行程序的效果如图 10-4 所示。

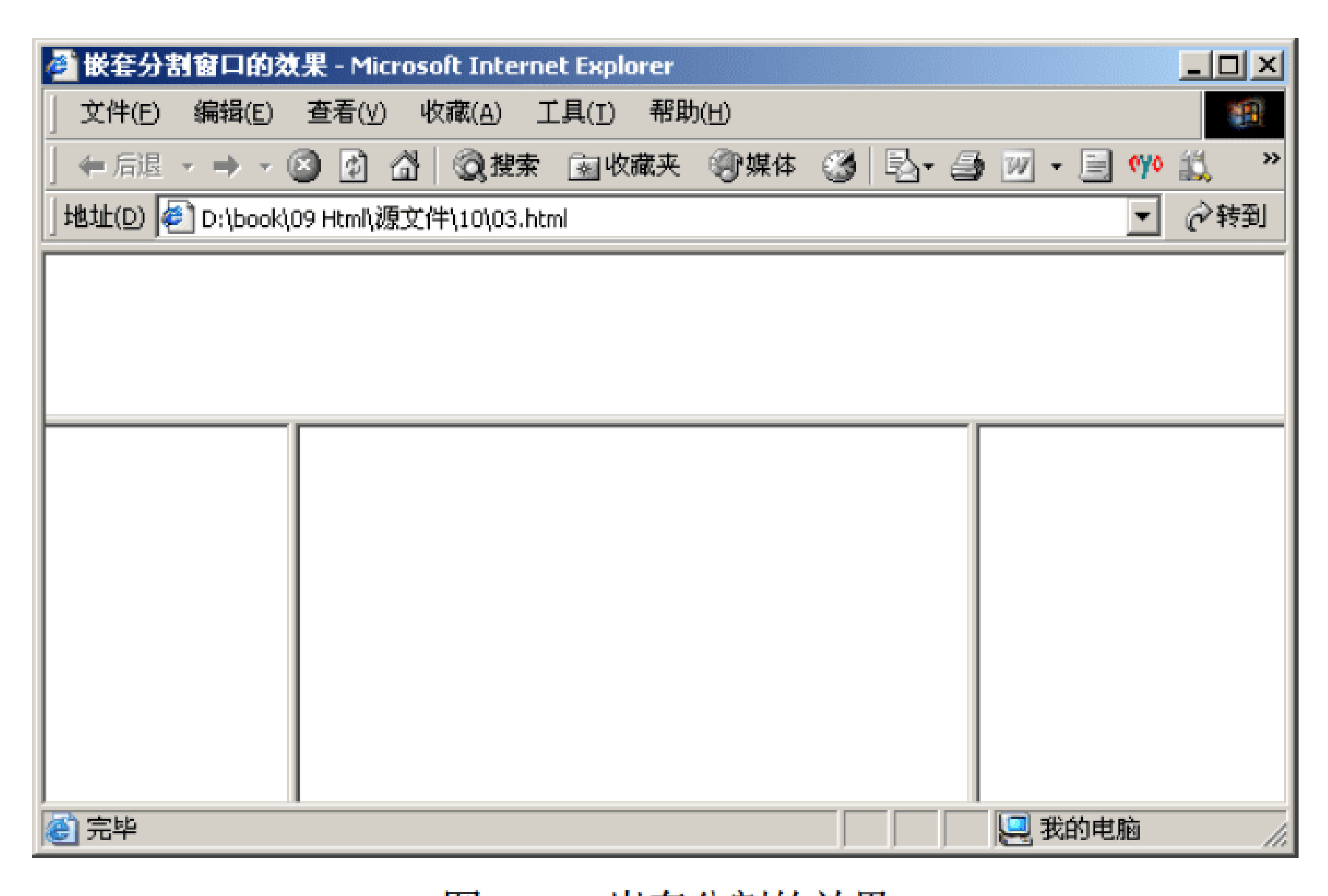


图 10-4　嵌套分割的效果

10.1.3　边框属性 frameborder

由前面的几个实例可以看出，在默认情况下，框架窗口的四周有一条边框线，通过 frameborder 参数可以调整边框线的显示情况。语法结构如下所示。

```
<frameset frameborder="属性值">其他框架元素与框架集</frameset>
```

frameborder 属性的取值，可以使用英文的“no”和“yes”，也可以使用数字“0”和“1”。其中“no”和“0”代表没有边框，“yes”和“1”代表有边框。

下面是一个使用 frameborder 属性的实例。其代码如下所示。

例程 10-5 frameset-frameborder.html

```
<frameset rows="30%,55%,15%">
 <frame frameborder="1">
   <frameset cols="25%,75%" frameborder="0">
      <frame >
      <frame>
   </frameset>
<frame frameborder="0">
</frameset>
</html>
```

该实例中，指定了第一个框架含有边框，第二个框架不含边框。其代码运行后，显示效果如图 10-5 所示。

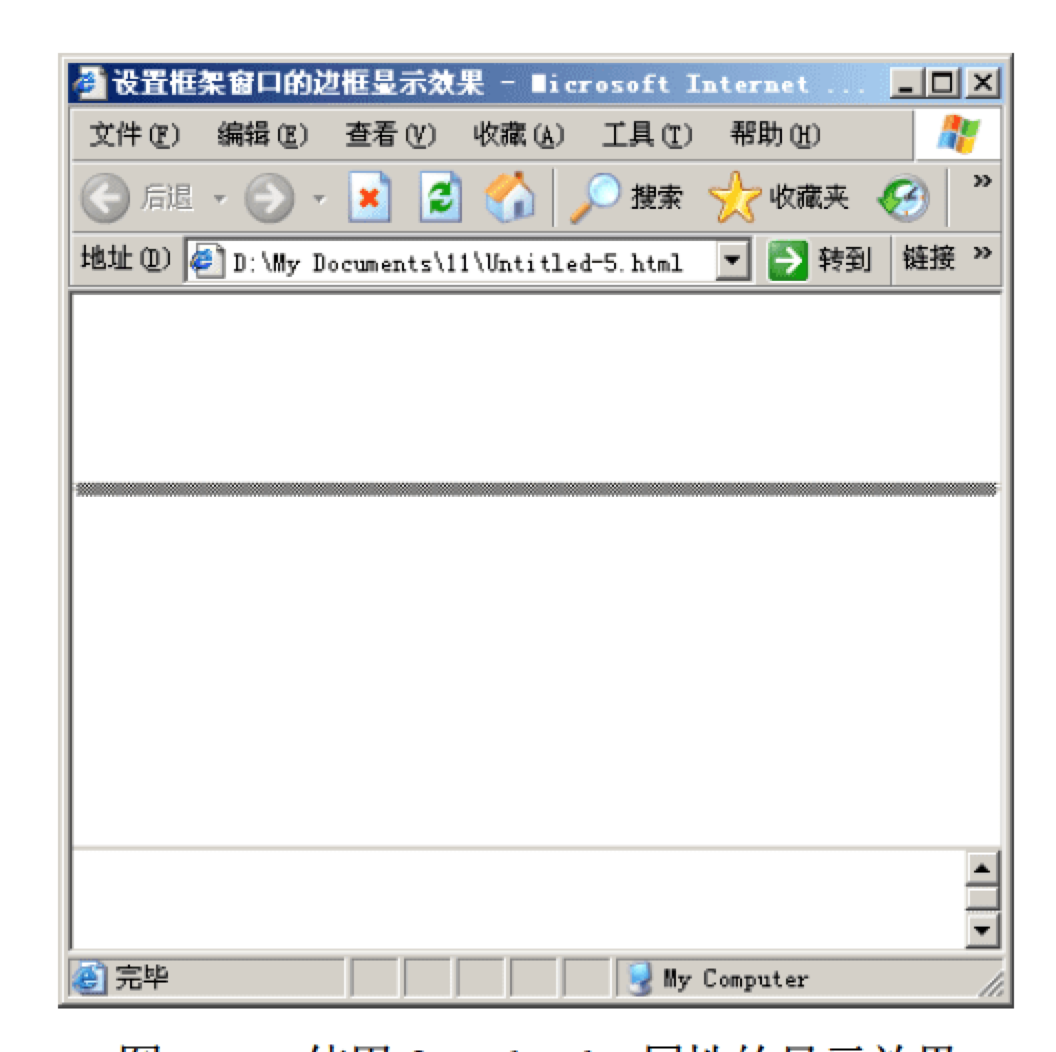


图 10-5 使用 frameborder 属性的显示效果

10.1.4 边框宽度属性 framespacing

框架的边框宽度在默认情况下是 1 像素，在 HTML 中用框架集元素的边框宽度属性 framespacing 来定义框架区域边框的宽度。framespacing 属性定义的边框宽度，在 IE 浏览器中是有效的，但是 Firefox 浏览器等不支持此属性。语法结构如下所示。

```
<frameset framespacing =" 边框宽度">其他框架元素与框架集</frameset>
```

边框宽度就是在页面中各个边框之间的线条宽度，以像素为单位。而这一参数只能对框架集使用，对单个框架无效。

下面是一个使用 framespacing 属性的实例。其代码如下所示。

例程 10-6 frameset-framespacing.html

```
<frameset rows="40%,60%" framespacing="10">
 <frame>
   <frameset cols="30%,45%,25%" framespacing="30">
      <frame >
      <frame >
      <frame >
   </frameset>
</frameset><noframes></noframes>
</html>
```

该实例中，指定了框架的边框宽度为 10。其代码运行后，显示效果如图 10-6 所示。

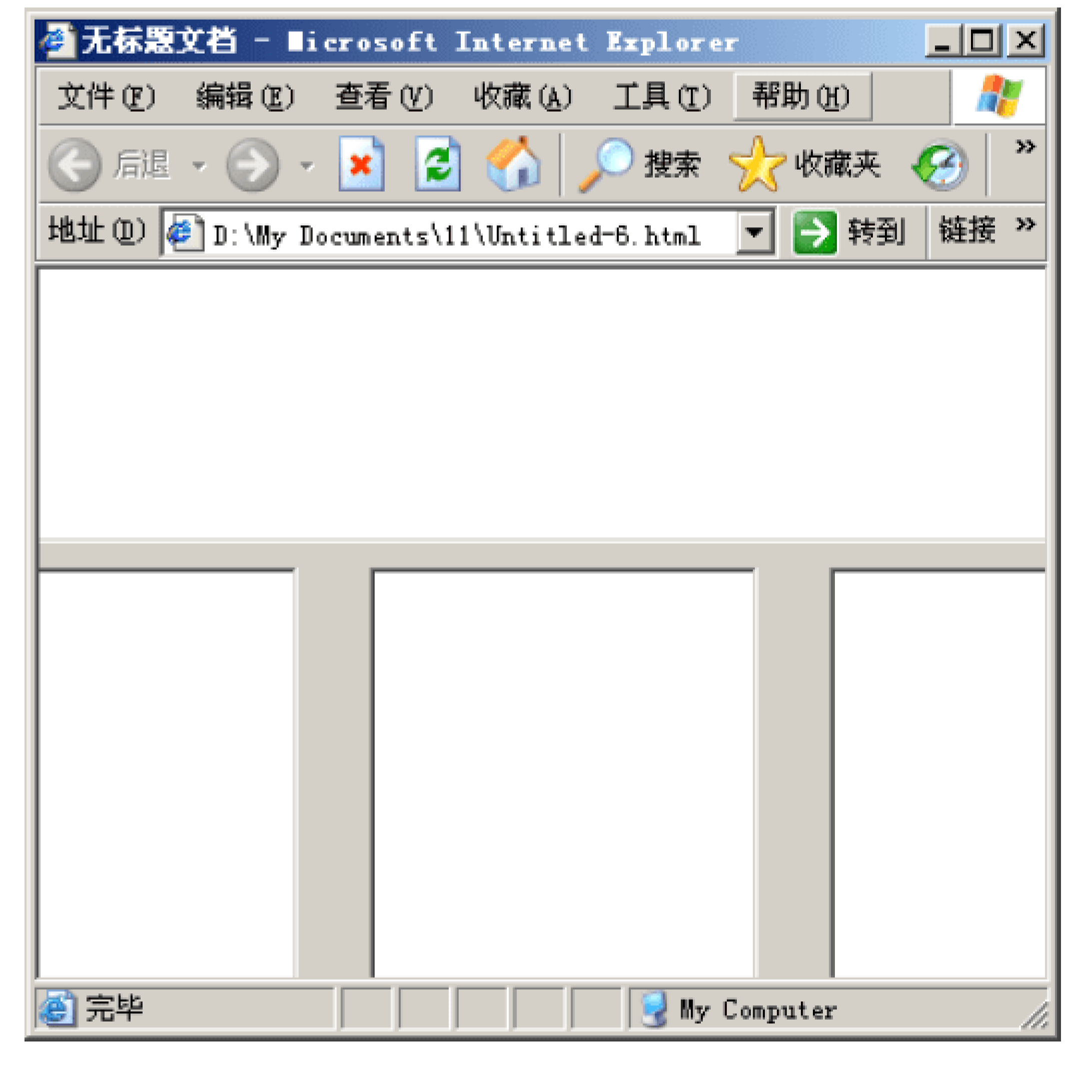


图 10-6　设置框架的边框宽度

10.1.5　边框宽度属性 border

框架集元素的边框宽度属性 border 用来定义框架区域边框的宽度。语法结构如下所示。

```
<frameset border ="数字值">其他框架元素与框架集</frameset>
```

其中数字值的实际单位是像素。下面是一个使用 border 属性的实例。其代码如下所示。

例程 10-7　frameset-border.html

```
01    <frameset cols="150,100" rows="100,50" frameborder="yes" border="20">
      <frame    >  <frame    >
      <frame    > <frame    >
02    </frameset>
03    </html>
```

该实例中，01 行中指定了框架的边框宽度为 20。其代码运行后，显示效果如图 10-7 所示。

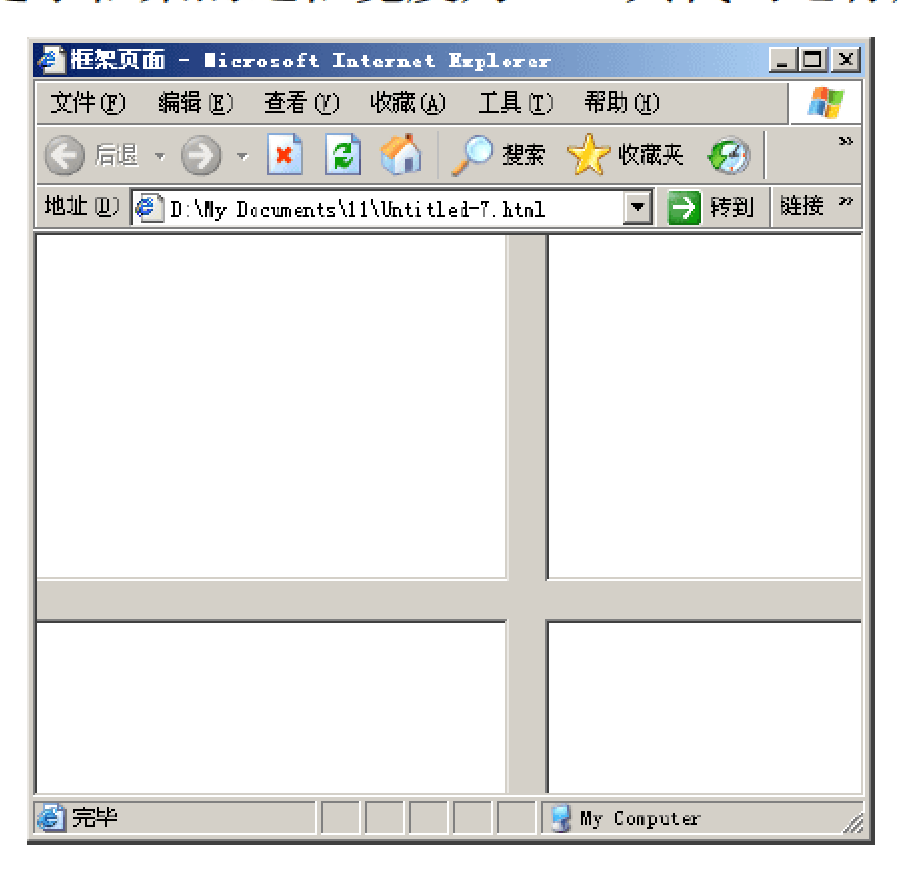


图 10-7　使用 border 属性的显示效果

如果同时使用 framespacing 属性和 border 属性，则在 IE 浏览器中会使用 framespacing 属性定义的值。为了在浏览器中显示一致的效果，最好为 framespacing 属性和 border 属性定义相同的值。

10.1.6 颜色属性 bordercolor

框架集元素的边框颜色属性 bordercolor 用来定义框架区域边框的颜色。语法结构如下所示。

```
<frameset bordercolor ="数字值">其他框架元素与框架集</frameset>
```

其中数字值的实际单位是像素。下面是一个使用 bordercolor 属性的实例。其代码如下所示。

例程 10-8 frameset-bordercolor.html

```
01  <frameset rows="40%,60%" framespacing="10" bordercolor ="#FF0000">
    <frame>
02  <frameset cols="30%,45%,25%" framespacing="30" bordercolor ="#9900FF">
     <frame >
     <frame>
     <frame>
03  </frameset>
04  </frameset><noframes></noframes>
05  </html>
```

该实例中，01 行指定了框架的水平边框宽度为 10，同时定义的框架边框的颜色为红色。02 行指定竖直边框宽度为 20，同时定义框架边框颜色为紫色。其代码运行后，显示效果如图 10-8 所示。

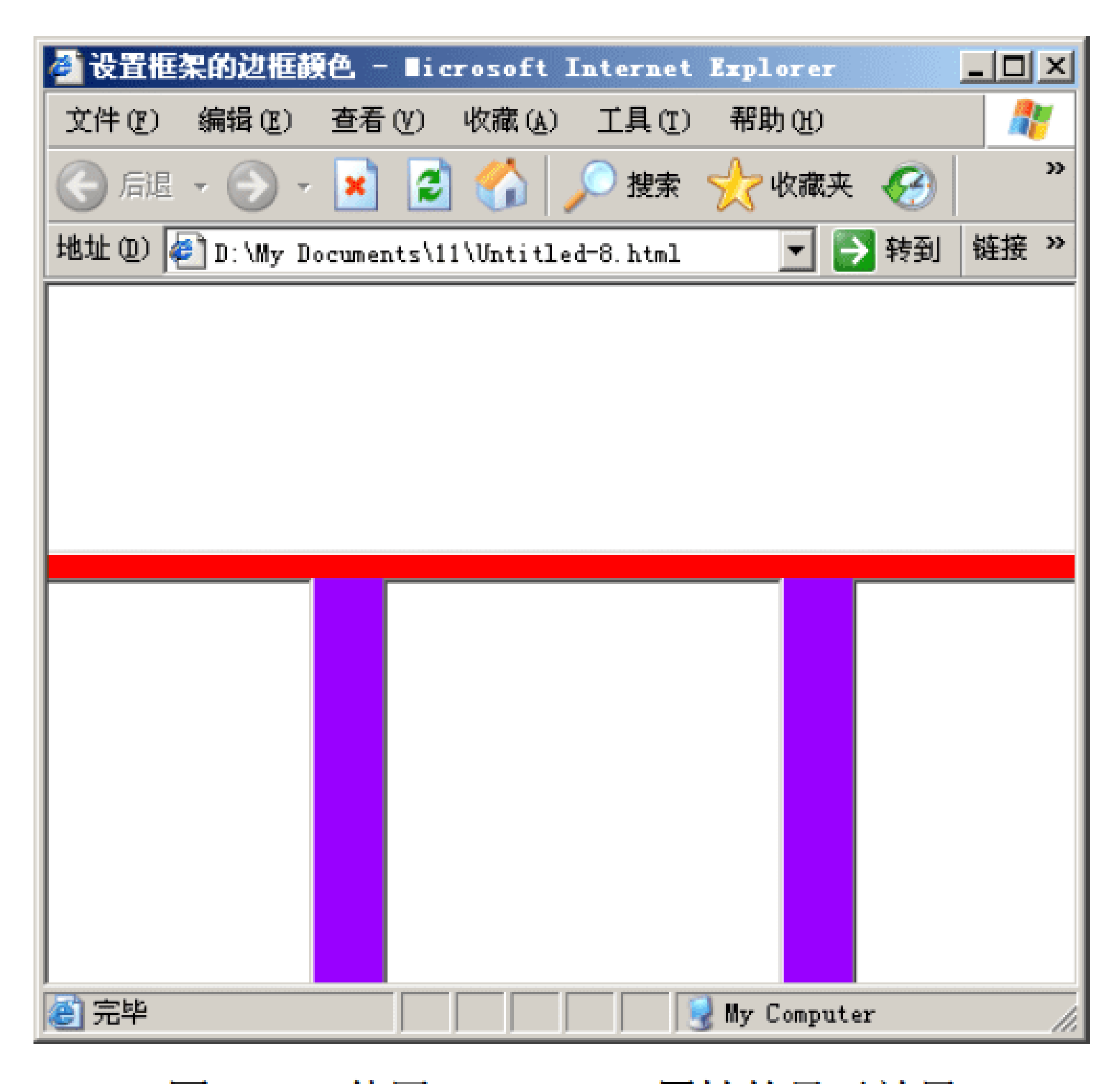


图 10-8 使用 bordercolor 属性的显示效果

10.2 框架内容元素<frame>

框架内容元素<frame>用来在页面中定义框架集的内容。其中每条内容会以从左至右、从上到下的顺序排列。语法结构如下所示。

```
< frameset >
    <frame  属性="属性值"/>
</frameset>
```

在网页中，<frame>元素要和<frameset>元素一起使用。下面是一个使用<frame>元素的实例。其代码如下所示。

例程 10-9　frame.html

```
01  <frameset cols="50,100" rows="100,50" frameborder="yes" border="0" framespacing="0" >
    <frame scrolling="yes" >
    <frame   scrolling="no" >
    <frame   scrolling="yes" >
    <frame scrolling="no"   >
02  </frameset>
03  </html>
```

该实例中，使用了 4 个<frame>元素，其中第 1 个和第 3 个定义了滚条属性。其运行后的显示效果，如图 10-9 所示。

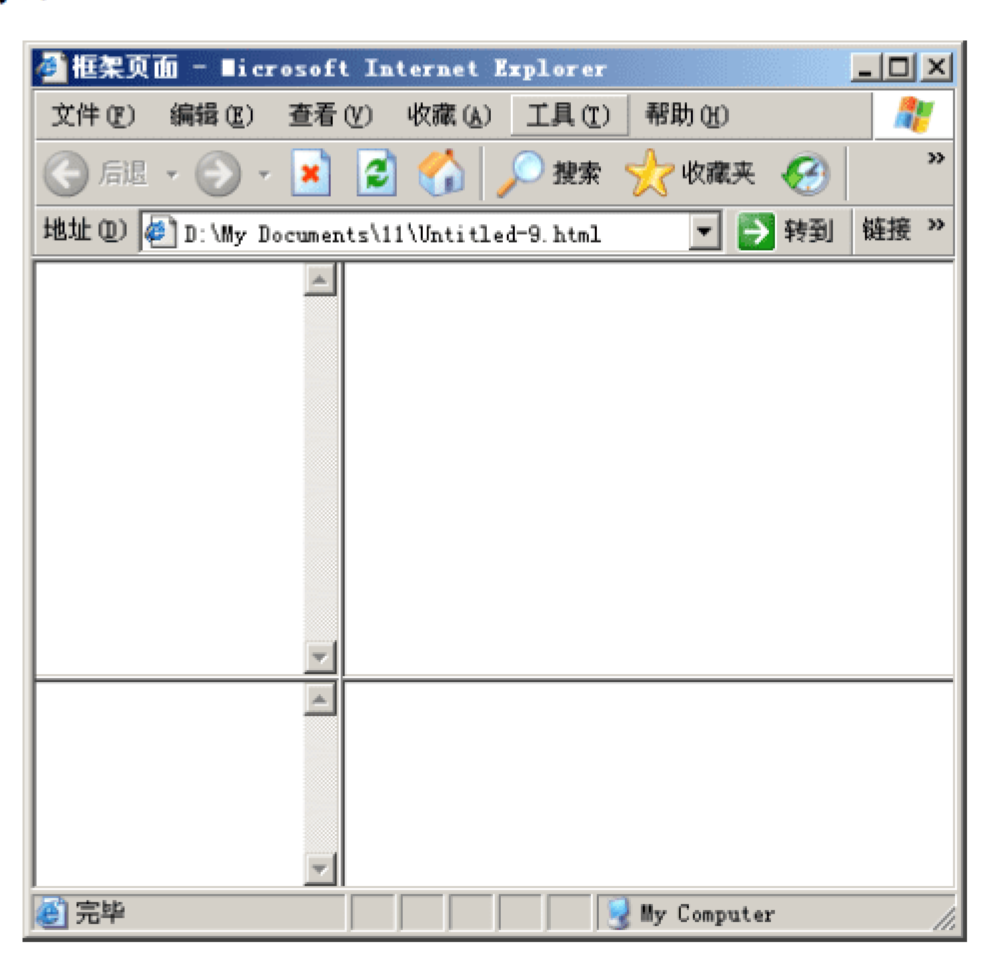


图 10-9　使用<frame>元素的显示效果

10.2.1　内容的路径属性 src

显示内容的路径属性 src 用来定义框架集内容的路径。语法结构如下所示。

```
<frame src="  页面源文件地址"/>
```

其中路径值既可以使用相对路径，也可以使用绝对路径。下面是一个使用 src 属性的实例。其代码如下所示。

例程 10-10　frame-src.html

```
<frameset frameborder="no" border="0" framespacing="0" >
<frame src="http://www.baidu.com" name="main" scrolling="yes" id="main" />
<frame ></frameset>
</html>
```

该实例中，使用了一个框架集，并在其中引用了百度的官方站点的地址。其运行后的显示效果，如图 10-10 所示。



图 10-10　使用 src 属性的显示效果

10.2.2　滚条属性 scrolling

滚条属性 scrolling 用来指定框架区域是否产生滚条。在框架页面中，如果不指定滚条的显示，则当内容大于显示区域时滚条会自动显示滚条。语法结构如下所示。

```
<frame src="路径" scrolling="属性值"/>
```

下面是一个使用 scrolling 属性的实例。其代码如下所示。

例程 10-11　frame-scrolling.html

```
<frameset cols="50,100" rows="100,50" frameborder="yes" border="10"
  framespacing="10" bordercolor="#000000">
<frame    scrolling="yes" >
<frame src="http://www.baidu.com" name="right_topframe" scrolling="auto" id="right_topframe" />
<frame >
<frame src="http://www.sina.com" name="right_bottomframe" scrolling="auto" id="right_bottomframe" />
</frameset>
</html>
```

该实例中，分别在各个框架区域中使用 scrolling 属性的各种取值，其中一个区域中没有定

义 scrolling 属性。其运行后的显示效果，如图 10-11 所示。



图 10-11 使用 scrolling 属性的显示效果

10.2.3 固定尺寸属性 noresize

固定尺寸属性 noresize 用来指定框架区域的大小不可以拖动。在框架页面中，如果固定某个区域的大小，则可以在页面中使用鼠标拖动框架来改变其大小。语法结构如下所示。

```
<frame src="路径" noresize="noresize"/>
```

因为 noresize 属性没有值，在 XHTML 中要使用本身的名称作为值。

1. 不使用 noresize 属性的实例

下面是一个不使用 noresize 属性的实例。其代码如下所示。

例程 10-12 frame-noresize.html

```
<frameset cols="50,100" rows="100,50" frameborder="yes" border="10"
framespacing="10" bordercolor="#000000">
    <frame src="left_top.html" name="left_topframe" scrolling="yes" id="left_topframe" />
    <frame src="http://www.baidu.com" name="right_topframe" scrolling="auto" id="right_topframe" />
    <frame src="left_bottom.html" name="left_bottomframe" id="left_bottomframe" />
<frame src="http://www.sina.com" name="right_bottomframe" scrolling="no" id="right_bottomframe" />
</frameset>
</html>
```

其运行后的显示效果，如图 10-12 所示。



图 10-12　不使用 noresize 属性的显示效果

2．使用 noresize 属性的实例

下面是一个使用 noresize 属性的实例。如果在上面实例的框架中，指定了 4 个区域中的任意一个区域的 noresize 属性，则其他区域的大小也不能够改变。其原因在于，4 个区域是相互联系的区域。其代码如下所示。

例程 10-13　frame-noresizetwo.html

```
<frameset cols="50,100" rows="100,50" frameborder="yes" border="10"
framespacing="10" bordercolor="#000000">
<frame src="left_top.html" name="left_topframe" noresize="noresize" scrolling="yes" id="left_topframe" />
<frame src="http://www.baidu.com" name="right_topframe" scrolling="auto" id="right_topframe" />
<frame src="left_bottom.html" name="left_bottomframe" id="left_bottomframe" />
 <frame src="http://www.sina.com" name="right_bottomframe" scrolling="no" id="right_bottomframe" />
</frameset>
</html>
```

其运行后的显示效果，如图 10-13 所示。

使用 noresize 属性后的效果，并不是固定了某个区域的尺寸，而是固定区域所占有的比例。

10.2.4　内容的显示位置属性 marginheight、marginwidth

内容的显示位置属性 marginheight、marginwidth 用来指定框架区域内的文本和区域边界之间的距离。其中，marginheight 属性用来指定垂直方向的距离，marginwidth 属性用来指定水平方向的距离。语法结构如下所示。



图 10-13　使用 noresize 属性的显示效果

```
<frame src="路径" marginheight ="数字值" marginwidth="数字值"/>
```

可以单独使用 marginheight 属性或者 marginwidth 属性，也可以同时使用两个属性。其中数字值的实际单位是像素。

1．使用 marginheight 属性的实例

下面是一个使用 marginheight 属性的实例。其代码如下所示。

例程 10-14　frame-marginheight.html

```
<frameset cols="50,100" rows="100,50" frameborder="yes" border="10"
framespacing="10" bordercolor="#000000">
<frame src="left_top.html" name="left_topframe" scrolling="no" id="left_topframe" />
<frame src="main.html" name="mainframe" id="mainframe" title="mainframe" />
<frame src="left_bottom.html" name="left_bottomframe" scrolling="no" id="left_bottomframe" />
<frame src="right_bottom.html" name="right_bottomframe" scrolling="no" id="right_bottomframe" />
</frameset>
</html>
```

该实例中，在 id 为“right_topframe”的区域中，定义了 marginheight 属性取值为 0，其运行后的显示效果，如图 10-14 所示。

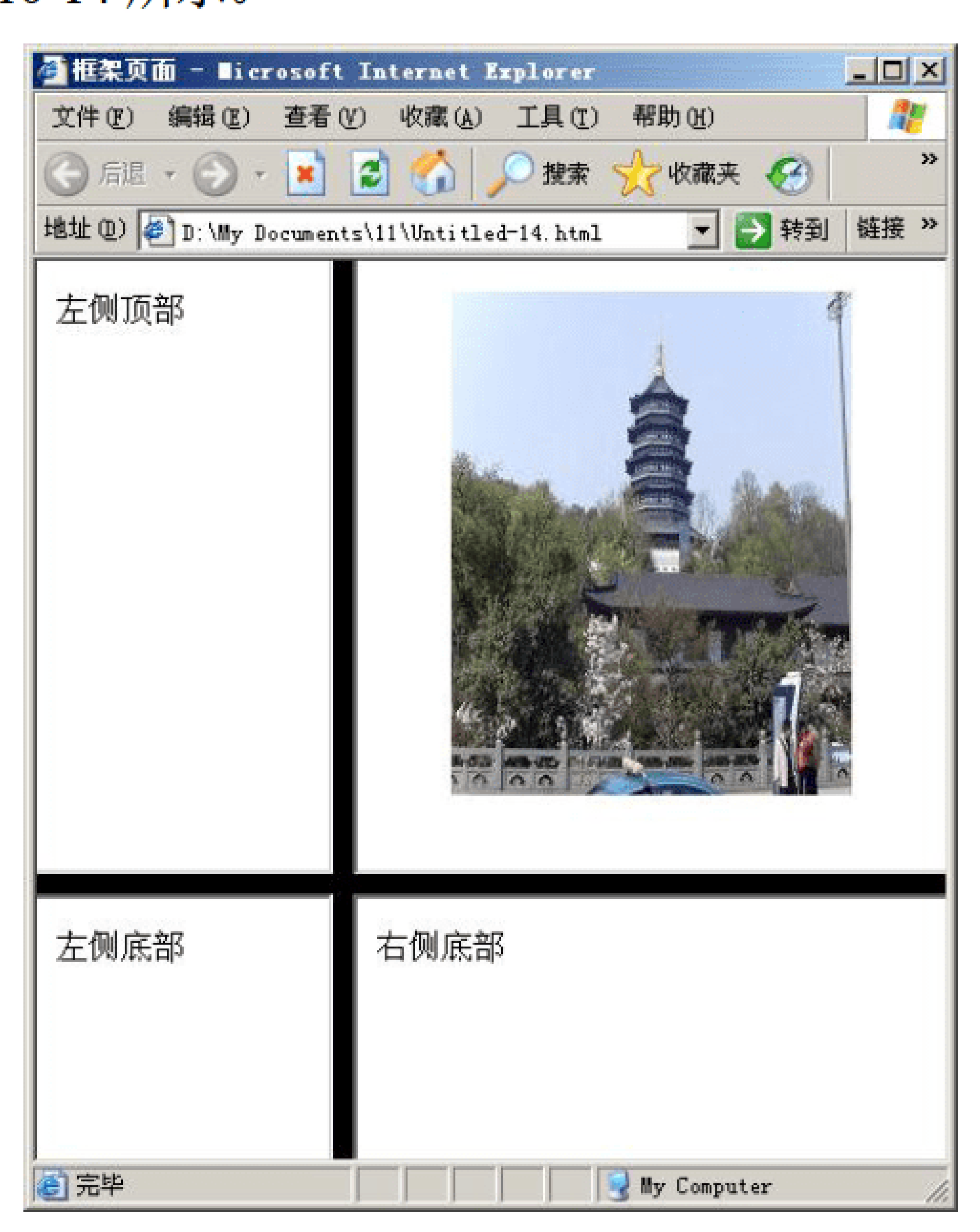


图 10-14　使用 marginheight 属性的显示效果

从图 10-14 中可以看到，当在 id 为“right_topframe”的区域中，定义了 marginheight 属性后，marginwidth 属性会默认的显示为 0。

2．使用 marginwidth 属性的实例

下面是一个使用 marginwidth 属性的实例。其代码如下所示。

例程 10-15 frame-marginwidth.html

```
<frameset cols="50,100" rows="100,50" frameborder="yes" border="10"
framespacing="10" bordercolor="#000000">
<frame src="left_top.html" name="left_topframe" scrolling="no" id="left_topframe" />
<frame src="right_top.html" name="right_topframe" scrolling="no" id="right_topframe" marginheight="0"
marginwidth="20" />
<frame src="left_bottom.html" name="left_bottomframe" scrolling="no" id="left_bottomframe" />
<frame src="right_bottom.html" name="right_bottomframe" scrolling="no" id="right_bottomframe" />
</frameset>
</html>
```

该实例中，在 id 为“right_topframe”的区域中，定义了 marginwidth 属性取值为 20，其运行后的显示效果，如图 10-15 所示。

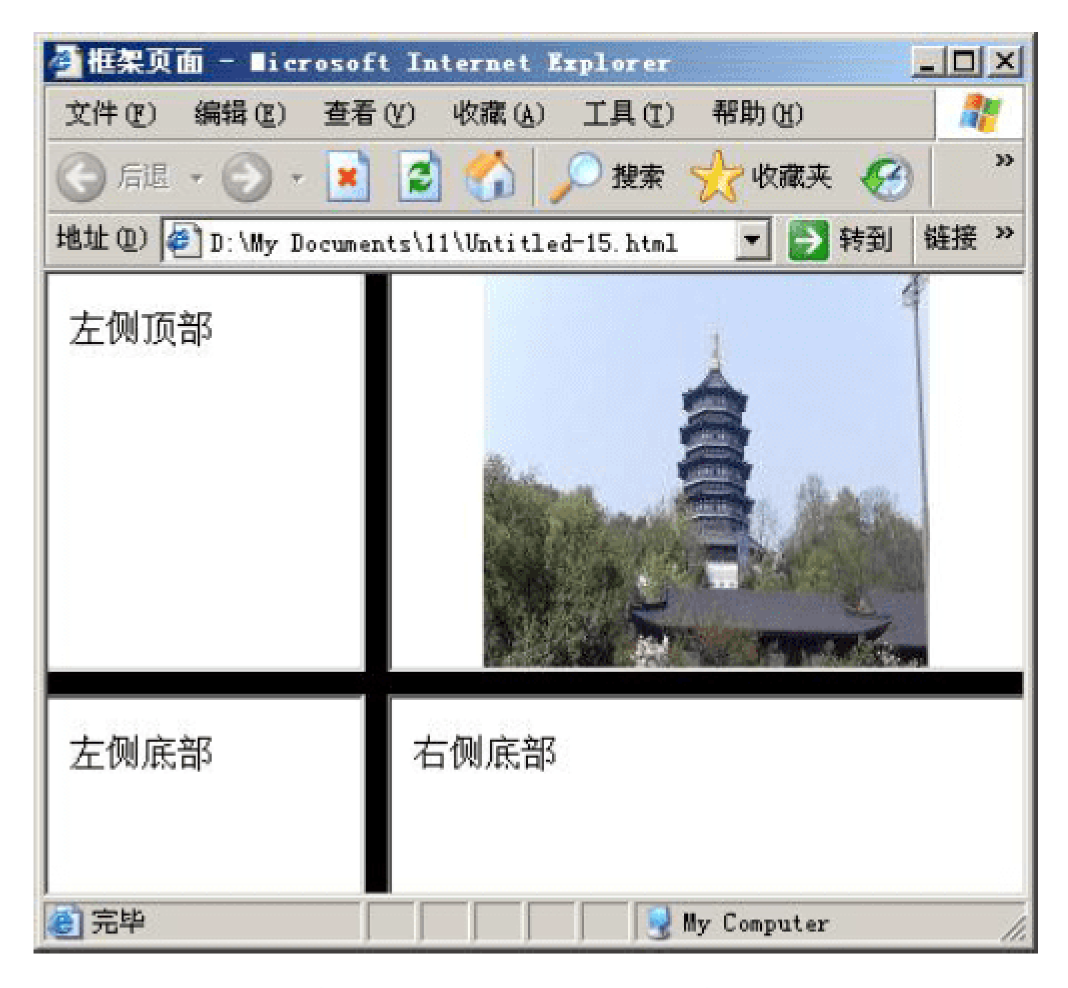


图 10-15 使用 marginwidth 属性的显示效果

从图 10-15 中可以看到，当在 id 为“right_topframe”的区域中，定义了 marginwidth 属性后，在水平方向上内容距离区域左边界的距离为 20 像素。

10.2.5 边框属性 frameborder

边框属性 frameborder 用来指定框架区域是否含有边框。语法结构如下所示。

```
<frame src="路径" frameborder ="属性值" />
```

frameborder 属性的取值，可以使用英文的“no”和“yes”，也可以使用数字“0”和“1”。其中“no”和“0”代表没有边框，“yes”和“1”代表有边框。下面是一个使用 frameborder 属性的实例。其代码如下所示。

例程 10-16 frameframeborder-.html

```
<frameset cols="50,100" rows="100,50" frameborder="no">
<frame src="left_top.html" name="left_topframe" scrolling="no" id="left_topframe" />
<frame src="right_top.html" name="right_topframe" scrolling="no" id="right_topframe" frameborder="yes"
```

```
/>
        <frame src="left_bottom.html" name="left_bottomframe" scrolling="no" id="left_bottomframe" />
        <frame src="right_bottom.html" name="right_bottomframe" scrolling="no" id="right_bottomframe" />
    </frameset>
</html>
```

该实例中，在 id 为“right_topframe”的区域中，定义了 frameborder 属性取值为“yes”，其运行后的显示效果，如图 10-16 所示。



图 10-16　使用 frameborder 属性的显示效果

10.3 不支持框架元素<noframes>

不支持框架元素<noframes>用来为不能显示框架的浏览器制作内容。因为有一些浏览器或者浏览设备中并不支持框架元素，此时将不能正常显示内容。使用<noframes>元素可以使此类浏览器中显示<noframes>元素中的内容。语法结构如下所示。

```
<html>
  <noframes>内容部分</noframes>
</html>
```

使用<noframes>元素，对支持框架的页面的显示效果并没有影响。



10.4 内联框架元素<iframe>

内联框架元素<iframe>用来在页面的任何位置定义一个框架区域。使用<frameset>元素只

能用替换页面主体元素<body>的方式来定义框架，但是使用<iframe>元素可以在页面主体元素<body>中任何可以使用内联元素(如<img>元素等)的地方定义框架。语法结构如下所示。

```
<iframes>内容部分</iframes>
```

下面是一个使用<iframe>元素的实例。其代码如下所示。

例程 10-17 iframe.html

```
<body>
这是一个使用内联框架的实例<br />
这是内联框架<iframe src="http://www.baidu.com" width="400" height="200"></iframe>
</body>
</html>
```

这里选用百度作为内联框的内容，其运行后的显示效果如图 10-17 所示。



图 10-17 使用<iframe>元素的显示效果

10.5 框架元素中的链接

在框架元素中使用链接，和普通的链接略有不同。在框架元素中，可以在链接中指定打开窗口的位置。从而实现在一个框架区域中，通过链接控制另一个框架区域的内容。下面进行详细的讲解。

通过在<a>元素中使用链接的目标属性 target，来定义打开链接的目标区域。语法结构如下所示。

```
<a href="路径" target="目标区域的 id">……</a>
```

在 target 属性的取值中，使用框架内容元素<frame>的标记 id，来指定链接打开的目标。下面是一个使用 target 属性的实例。其框架页面的代码如下所示。

例程 10-18 framelink.html

```
<frameset rows="*,80" cols="*" frameborder="yes" border="10" bordercolor="#000000"
framespacing="10">
    <frameset cols="*,150" frameborder="yes" border="10" bordercolor="#000000" framespacing="10">
        <frame src="main.html" name="mainframe" id="mainframe" title="mainframe" />
        <frame src="right.html" name="rightframe" scrolling="no" noresize="noresize" id="rightframe"
title="rightframe" />
</frameset>
<frame src="bottom.html" name="bottomframe" scrolling="no" noresize="noresize" id="bottomframe"
title="bottomframe" />
</frameset><noframes></noframes>
</html>
```

在框架页面中，定义了所有框架内容的 id 属性。在每个区域的调用页面中，就可以使用相应的 id 属性值来确定链接的打开目标。其中含有链接的“right.html”页面的代码如下所示。

例程 10-19 right.html

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>无标题文档</title>
<style>
body{
background:#666666;
font-size:18px;
color:#ffffff;
font-weight:bold;}
a{
  color:#ffffff;}
</style>
</head>
<body>
<ul>
<li><a href="http://www.baidu.com" target="mainframe">百度</a></li>
<li><a href="http://www.sina.com.cn" target="mainframe">新浪</a></li>
<li><a href="http://www.sohu.com" target="mainframe">搜狐</a></li>
<li><a href="http://www.yahoo.com" target="mainframe">雅虎</a></li></ul>
</body>
</html>
```

其中框架页面运行后的显示效果，如图 10-18 所示。



图 10-18　框架页面的原始的显示效果

当单击右侧的某个链接时，新的页面将在左侧含有老虎图片的区域中打开。单击其中的链接时，就会显示这些页面。

10.6 本章习题

一、选择题

1．要创建一个如图 10-19 所示的框架，应使用以下代码：(　　)

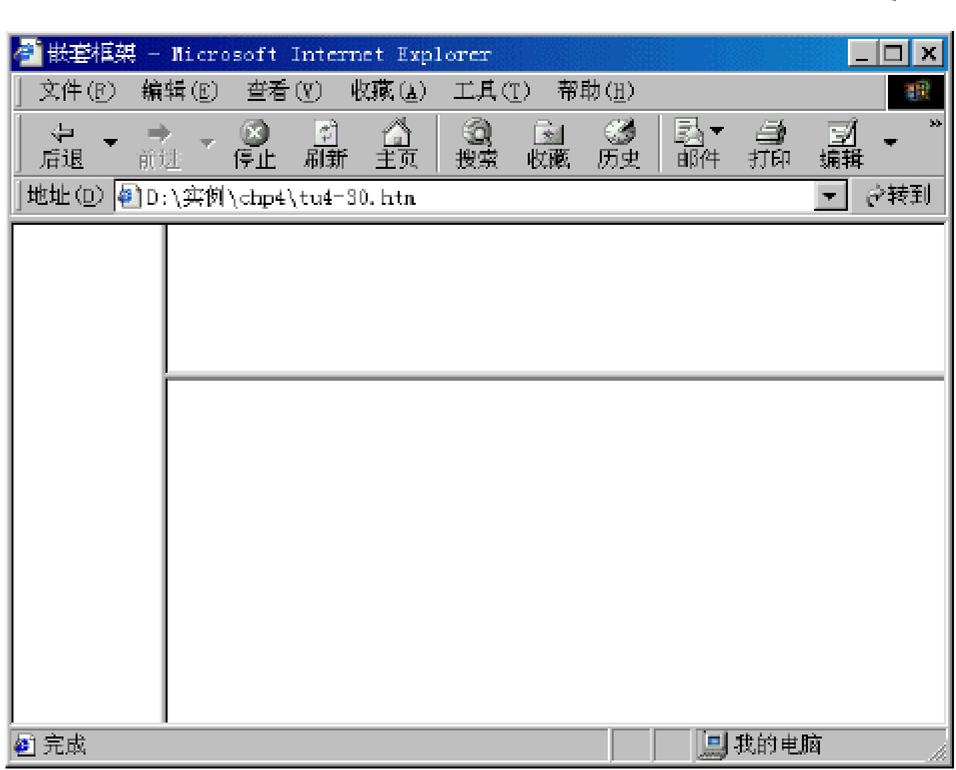


图 10-19　框架

A. <FRAMESET cols="100,* ">

<FRAME>

<FRAMESET cols="120,*">

<FRAME>

<FRAME>

</FRAMESET>

```
</FRAMESET>
B.<FRAMESET rows="100,* ">
    <FRAME>
    <FRAMESET cols="120,*">
        <FRAME>
        <FRAME>
    </FRAMESET>
</FRAMESET>
C.<FRAMESET cols="100,* ">
    <FRAME>
    <FRAMESET rows="120,*">
        <FRAME>
        <FRAME>
    </FRAMESET>
</FRAMESET>
D.<FRAMESET rows="100,120,* ">
    <FRAME>
    <FRAME>
    <FRAME>
</FRAMESET>
</FRAMESET>
```

2．要创建一个如图 10-20 所示的框架，应使用以下代码：(　　)

```
A. <FRAMESET cols="100,* ">
    <FRAME>
    <FRAMESET cols="150,*">
        <FRAME>
        <FRAME>
    </FRAMESET>
</FRAMESET>
B.<FRAMESET rows="100,* ">
    <FRAME>
    <FRAMESET cols="150,*">
        <FRAME>
        <FRAME>
    </FRAMESET>
```

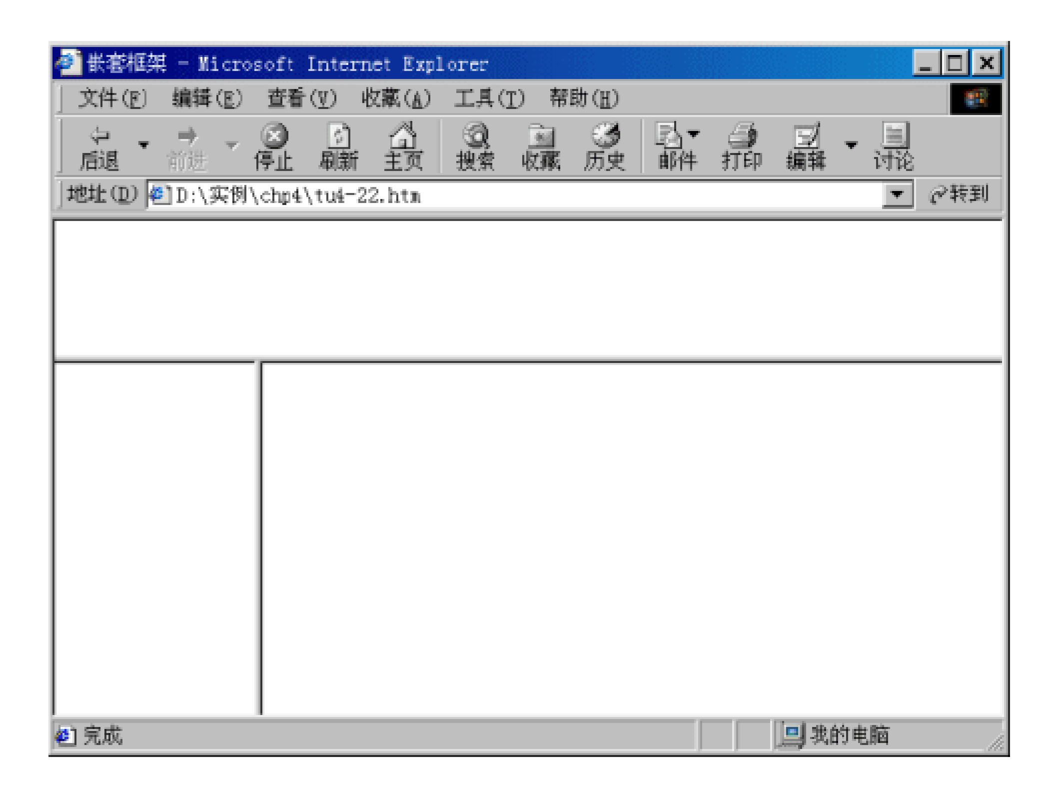


图 10-20　框架

```
</FRAMESET>
C.<FRAMESET cols="100,* ">
  <FRAME>
  <FRAMESET rows="150,*">
    <FRAME>
    <FRAME>
  </FRAMESET>
</FRAMESET>
D.<FRAMESET rows="100,120,* ">
  <FRAME>
  <FRAME>
  <FRAME>
</FRAMESET>
</FRAMESET>
```

3．以下关于框架显示效果的说法中，错误的是：(　　)

A. 只有所有相邻框架的边框都设置为 0，才能隐藏边框。

B. 可以在 FRAME 标记符中使用 marginwidth 和 marginheight 属性控制框架内容与框架边框之间的距离。

C. 框架的边框默认可以移动。

D. 框架默认时有滚动条。

4．有关框架与表格的说法正确的有：(　　)。

A. 框架对整个窗口进行划分　　B. 每个框架都有自己独立网页文件

C. 表格比框架更有用　　D. 表格对页面区域进行划分

5．在一个框架的属性面板中，不能设置下面哪一项。(　)

A. 源文件　　B. 边框颜色　　C. 边框宽度　　D. 滚动条

二、填空题

1．在一个分为左右两个框架的框架组中，要想使左侧的框架宽度不变，应该用______单位来定制其宽度，而右侧框架则使用___单位来定制。

2．框架主要有____________和____________两部分组成。

3．框架中用________属性来指定行的数目，用______来指定列的数目。

4. 把一个页面分成 30%、20%、50%的 3 列代码写法是(写出分列代码即可)：______________

5. 把一个页面分成 40%、60%的 2 行代码写法是：______________

6．在一个 2 行 3 列的框架中，设置水平边框宽度为 10。垂直边框宽度为 20 的代码写法是：____________

三、实战练习

已知网页初始效果如图 10-21 所示，整个窗口分为左右两框，左边框架为 150 像素；单击左边框架中的“文件 1”超链接，将在右边框架中显示“内容 1”；单击左边框架中的“文件 2”超链接，将在当前整个窗口中显示“内容 2”(即框架结构消失)。所有的超链接均没有下划线。请填写以下源代码中的空白。注意：所有 5 个文件都位于同一目录下。

```
<HTML><HEAD><TITLE>框架结构</TITLE></HEAD>
<FRAMESET ____=______>
<FRAME src="content.htm">
<FRAME src="main.htm" name=______>
</FRAMESET>
</HTML>

---------------content.htm-------------------
<HTML>
<HEAD><TITLE>目录</TITLE>
<STYLE>________
{text-decoration:none}</STYLE>
</HEAD>
<BODY>
<A href=file1.htm target="content">文件 1</A><P>
<A href=file2.htm target=________ >文件 2</A>
</BODY>
</HTML>

---------------main.htm---------------------
<HTML>内容</HTML>

----------file1.htm------------
<HTML>内容 1</HTML>

----------file2.htm------------
<HTML>内容 2</HTML>
```

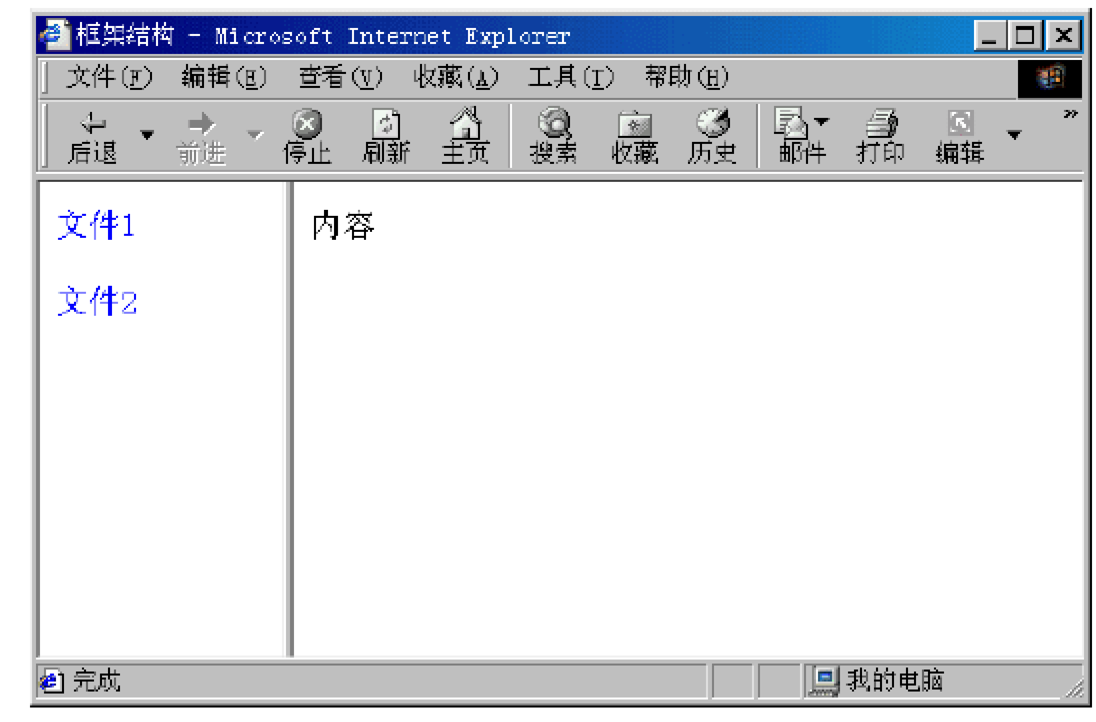


图 10-21　网页初始效果

Chapter 11

CSS的概念

在网页设计中，需要使用各种不同的元素和内容。为了页面更加美观，通常会给页面添加一些背景、边框之类，使页面的主题更加突出。在网页设计中，CSS 部分就是担任这个角色，其具体的实现过程，要依附于其他的相关代码。

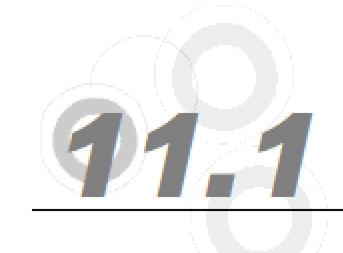
本章主要内容有:

◎ 网页设计中使用的各种内容。

◎ 在网页中简单使用 CSS 的方法。

◎ 调用 CSS 的方法。





11.1 什么是 CSS

CSS 又称为级联样式表，是 Cascading Style Sheet 的缩写，通常也简称为样式表。是 W3C 组织制定的，用于控制网页样式的一种标记性语言。使用 CSS 能使网页看起来更加美观漂亮。下面是一个应用 CSS 的页面的实例。其具体代码如下所示。

例程 11-1 example.html

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>CSS 样式</title>
<style>
<!--这里是 CSS 样式表开始-->
body{
margin:1;
padding:1;}
.content{
margin:60px auto;
height:150px;
width:350px;
 border:8px solid red;
 background:yellow;
line-height:150px;
text-align:center;
font-size:24px;}
<!--这里是 CSS 样式表结束-->
</style>
</head>
<body>
<!—这里是元素引用 css 样式的代码-->
<div class="main">
<div class="content">
CSS 样式</div></div>
</body>
</html>
```

代码中<style>元素所包含的部分就是样式表的部分。引用样式所使用的元素是 class 属性。应用样式后的页面效果如图 11-1 所示。如果取消定义的样式表，则显示效果如图 11-2 所示。

从图 11-1 和图 11-2 可以看到，取消了 CSS 后页面只剩下了内容部分，所有包括背景、字体样式、高度等修饰部分都消失了。

从实例中可以看出，CSS 的作用就是控制页面中内容部分的表现。通过使用 CSS 布局可以使得网站外观更加美观、结构更加清晰。使用样式表来实现页面表现的控制，是 W3C 制定的 Web 标准中推荐使用的方法。

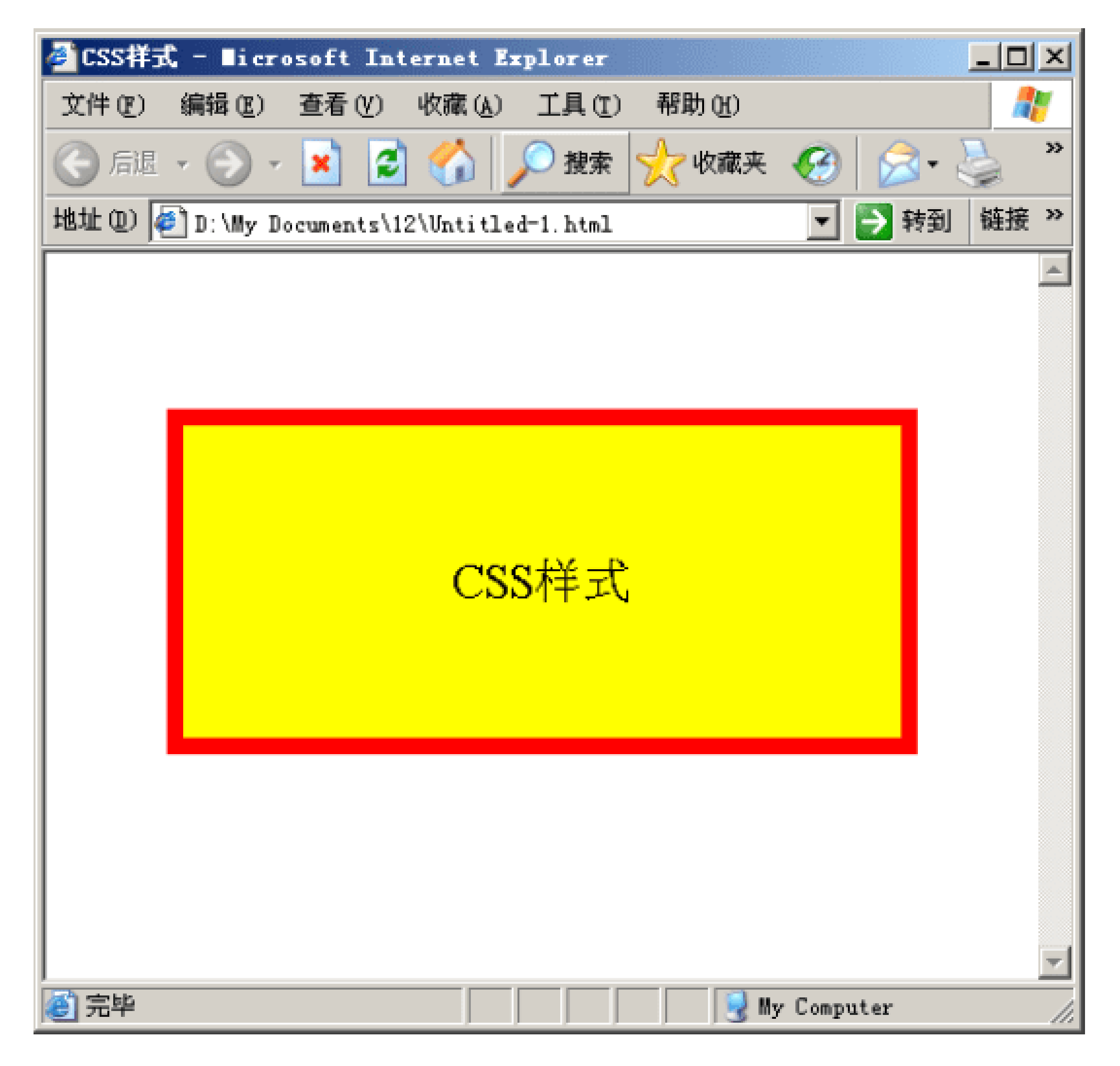




图 11-1　使用 CSS 样式的显示效果

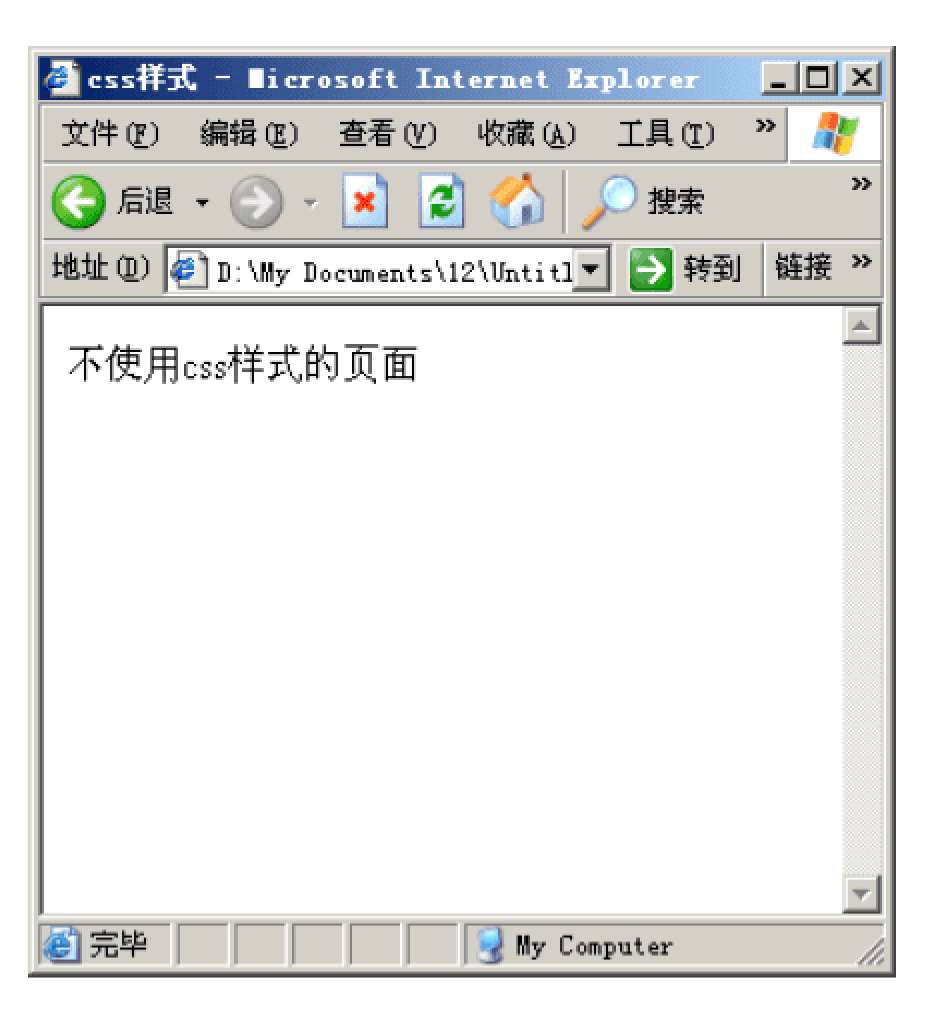


图 11-2　取消 CSS 样式的显示效果

11.2 CSS 与网页显示效果的关系

使用 CSS 对页面进行布局，可以避免大量冗余和重复的代码，同时使页面的更新和维护更加方便。CSS 布局之所以能够具有这样的优势，其原因在于：使用 CSS 布局的页面，结构和表现可以独立控制。通过修改 CSS 文件中定义的样式，可以统一修改站点中所有页面中相同的样式。同时，由于页面中剥离了修饰内容，减少了大量的冗余代码，使代码更加简洁清晰。由于 CSS 布局其清晰的结构、简洁的代码、高效的浏览速度，使得 CSS 布局的页面，对浏览者和网站拥有者都有极大的好处，具体表现在以下两个方面。

1．网站浏览者的好处

- 页面更加简洁，浏览速度更快，节省了浏览页面的时间。
- 因为代码的语义结构更加清晰并具有扩展性，使得页面能够被更多的用户，更广泛的浏览设备所支持。
- 结构和表现的分离，使用户可以独立选择界面样式，页面的显示效果更加灵活，例如现在的博客系统等。
- 通过的独立的 CSS 样式，更加便于控制页面的打印。

2．网站拥有者的好处

- CSS 布局时使用了清晰的语义结构，便于站点的更新和升级。
- 对于流量很大的站点，使用 CSS 布局，可以更大限度地压缩页面的大小，大大降低了成本。

➢ CSS 布局的页面更容易被搜索引擎辨识，方便站点的推广。
➢ 通过编写不同的 CSS 样式，可以在不影响内容的情况下，对网页进行任意改版。

由于 CSS 布局的页面带来的好处越来越明显，使得互联网中许多传统布局的站点都改为使用 CSS 布局。CSS 布局也将会是网页布局的发展趋势。

随着 Web 标准的提出，以及互联网对规范化的要求越来越高，大量的已有网站和新建网站都采用了 CSS 进行布局。使用 CSS 进行网页布局，不但可以使代码更加简洁，增加代码的可重用性，同样也可以制作出非常精美的页面。下面是一些使用 CSS 进行页面布局的站点，从页面的各种表现效果中，可以看出 CSS 控制页面表现的强大优势。

1. http://www.alexcohaniuc.com

该站点使用左右两栏的布局，运用简洁干净的色彩，明确简单的布局达到一种一目了然的效果。在站点页面中使用了各种修饰的内容，使页面在简洁的前提下又不失丰富，其显示效果如图 11-3 所示。

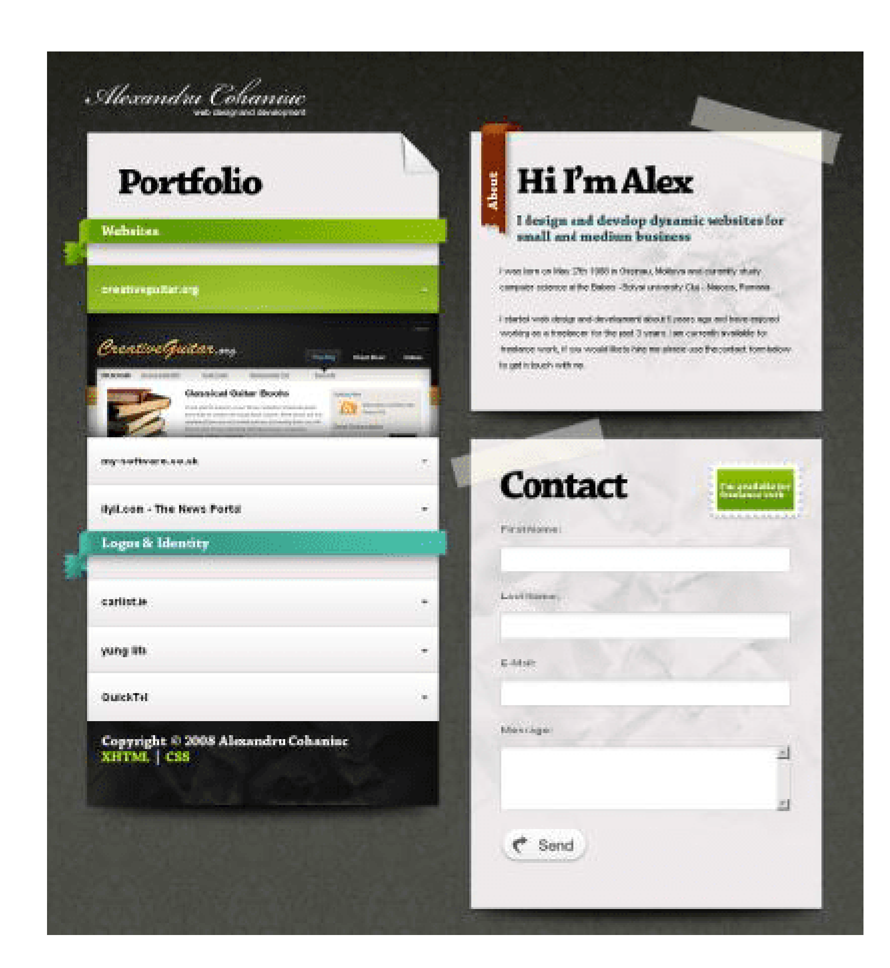


图 11-3　http://www.alexcohaniuc.com

2. http://www.vitiligo.com.br

该站点使用横向两栏，纵向三栏的布局方式，使页面的内容显得更加丰富。同时使用圆弧装的分隔线，对页面进行分隔，使页面在简洁中富有变化，其显示效果如图 11-4 所示。

图 11-4　http://www.vitiligo.com.br

3．http://nb.bottledsky.com

该站点使用纵向三栏的布局方式，同时在页面的顶部和底部使用渐变的背景图片，实现在不同的分辨率下都能够显示 100%的宽度。在实际制作页面的时候，使用重复的背景图片，适应页面宽度变化的方法，经常被采用，其显示效果如图 11-5 所示。

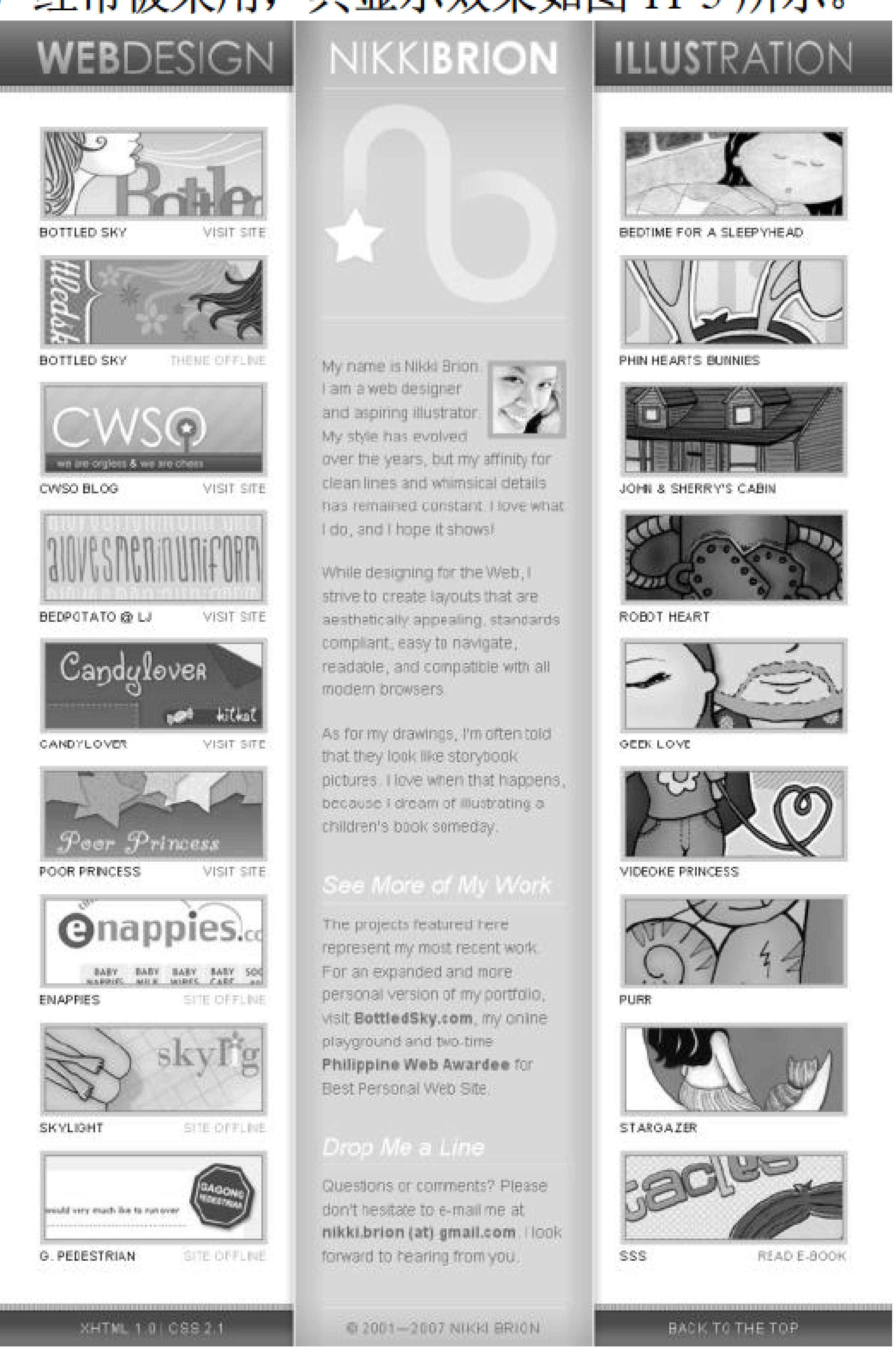


图 11-5　http://nb.bottledsky.com



11.3 使用 CSS 的方法

在页面中使用 CSS 样式，通常有 3 种方法：元素中直接添加样式、从页面头部<style>元素中调用、采用链接的形式调用。每种方法的添加位置和格式都有所区别，并且具有不同的优先级，其中推荐使用的方法是使用链接的外部样式。

11.3.1　元素中直接添加样式

在行内添加 CSS，就是将 CSS 样式添加在页面的元素标签中，此时样式的作用范围为元素标签开始和结束之间。行内添加 CSS 样式，具有最高的优先级。

下面是一个使用行内添加 CSS 样式的实例，其代码如下所示。

例程 11-2　example.html

```
<html>
<head>
<title>文档标题部分</title>
</head>
<body>
<div style="font-size:36px;line-height:100px;">这是一个使用内容样式的实例。</div>
</body>
</html>
```

以上的代码，使用 style 属性将 CSS 样式添加在<div>元素中。其中 CSS 样式的含义为：定义字体大小为 36 像素，行高为 100 像素。代码在运行后，显示效果如图 11-6 所示。

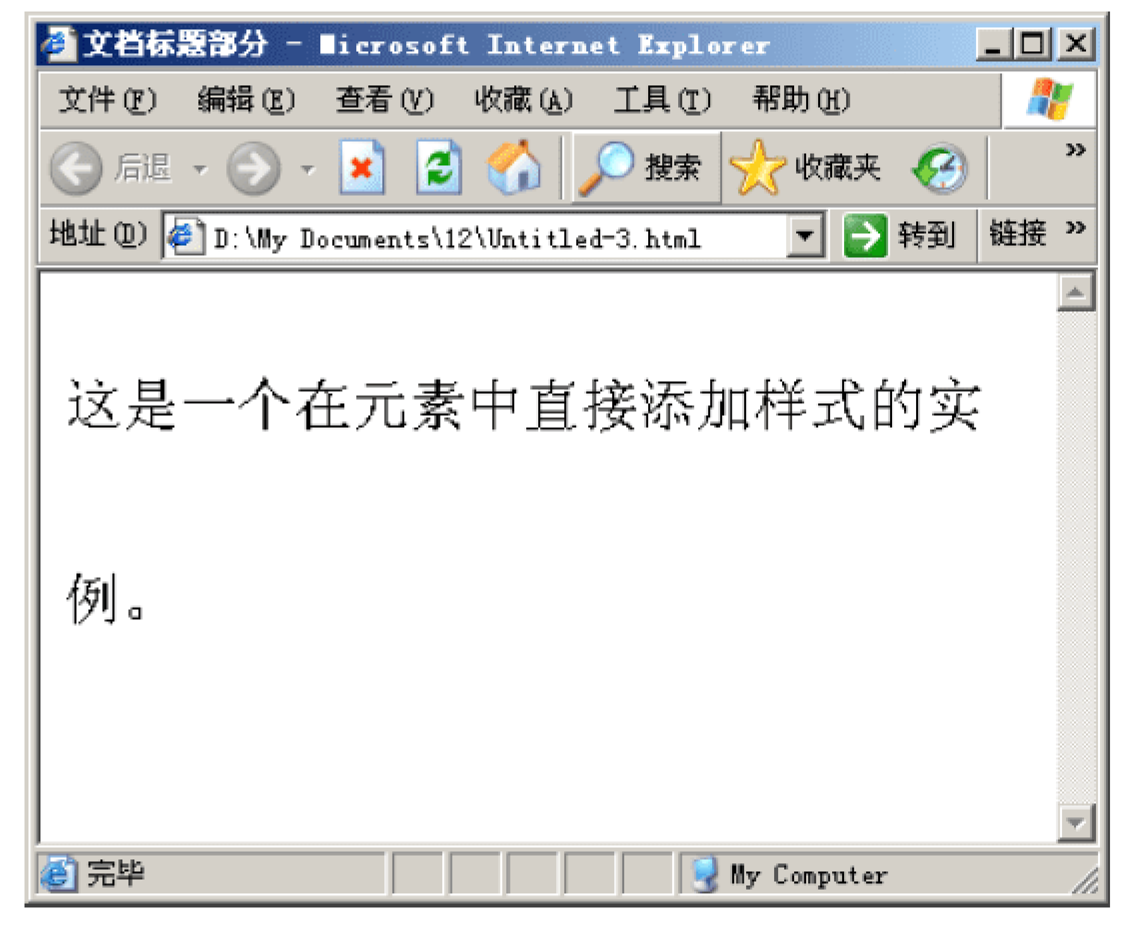


图 11-6　行内添加 CSS 样式后的显示效果

11.3.2　从页面头部<style>元素中调用

内嵌式调用 CSS 样式，就是通过在<style>元素中定义 CSS 代码，页面中调用相应样式的方式定义 CSS 样式。在<style>中定义的 CSS 样式必须通过页面中元素的名称或定义相应的属性才能够调用。内嵌式调用 CSS 样式，优先级低于行内添加的 CSS 样式。

下面是一个使用内嵌式调用 CSS 样式的实例，其代码如下所示。

例程 11-3　example.html

```
<html >
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>文档标题部分</title>
<style>
div{
font-size:24px;
line-height:150px;
color:block;
background-color:yellow;
```

```
}
</style>
</head>
<body>
 <div>这是从页面头部元素中调用 css 的实例。</div>
</body>
</html>
```

以上的代码中，通过在<head>元素中定义<style>元素，完成 CSS 样式的定义。其中定义的 CSS 样式除了定义字体大小和行高之外，还定义了字体的颜色和背景颜色。代码在运行后，显示效果如图 11-7 所示。

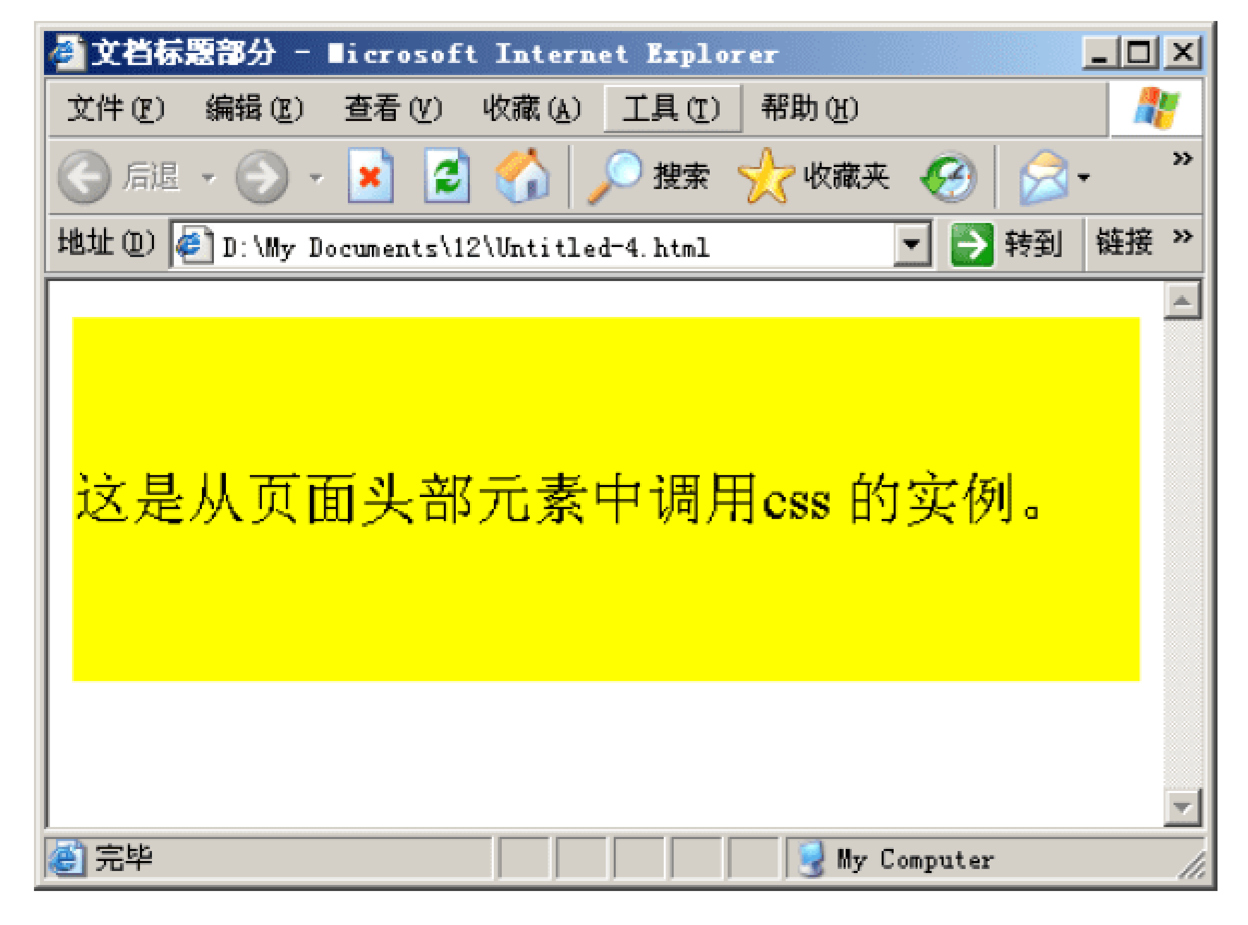


图 11-7　内嵌 CSS 样式后的显示效果

11.3.3　采用链接的形式调用

链接形式调用 CSS 样式，就是将 CSS 样式定义在一个外部的文件中，然后通过使用<link>元素，在页面中调用这个外部文件。独立的 CSS 文件后缀为“.css”为文本格式，可以保存在磁盘的任意位置。链接外部的 CSS 样式，优先级最低。

下面是一个链接外部 CSS 样式的实例，其代码如下所示。

例程 11-4　example.html

```
<html>
<head>
<title>文档标题部分</title>
<link href="css.css" type="text/css" rel="stylesheet" />
</head>
<body>
<div>这是一个使用内容样式的实例。</div>
</body>
</html>
```

以上的代码，使用<link>元素，链接了一个名称为“css.css”的外部样式文件。这个外部

样式文件中定义的 CSS 代码如下所示。

```
01 div{
    font-size:36px;
    line-height:100px;
    color:white;
    background-color: #999999;
    border:15px solid gray;
}
```

在这段 CSS 代码中，除了定义字体大小、行高、字体的颜色和背景颜色之外，还定义了元素的边框为 15px 的灰色实线边框。代码在运行后，显示效果如图 11-8 所示。

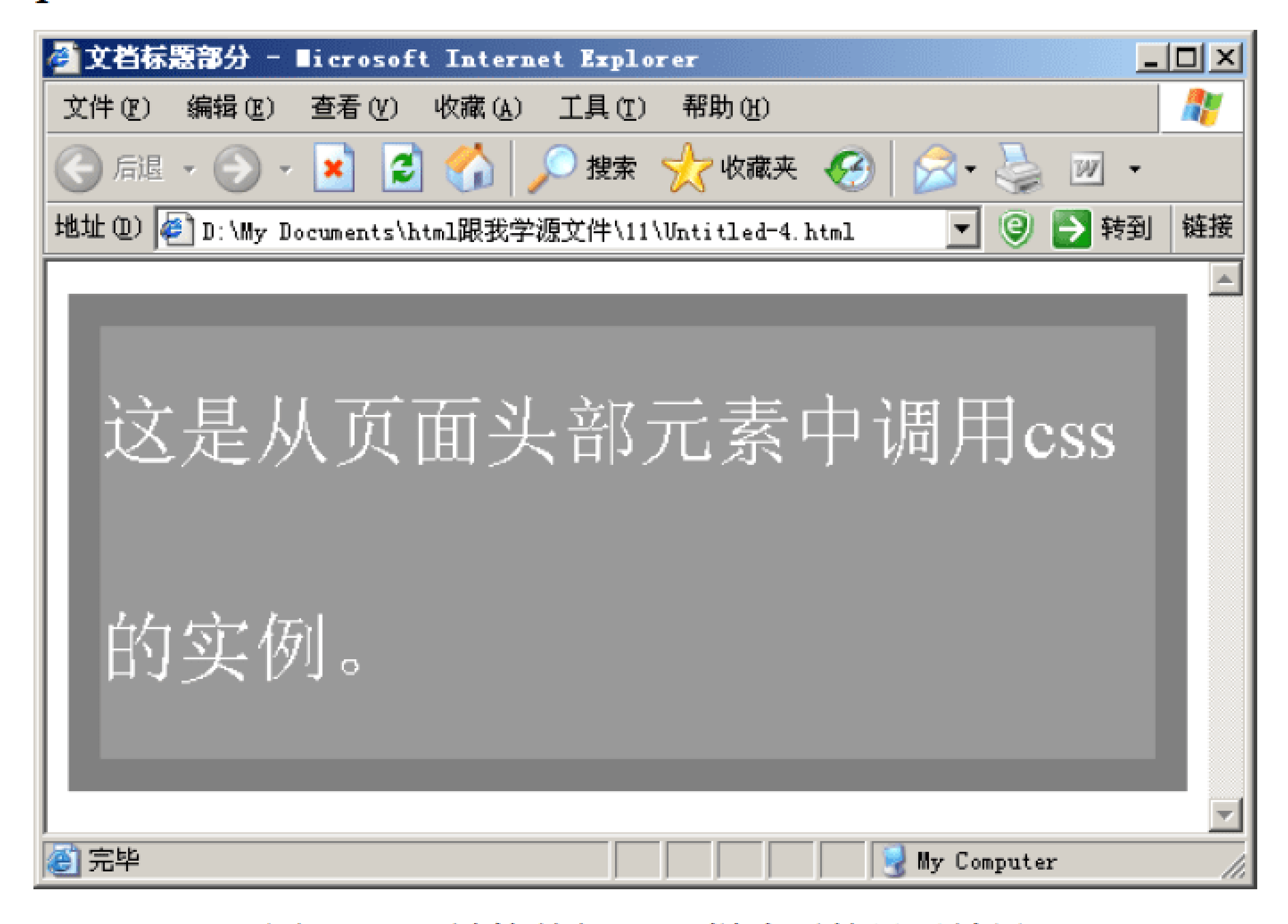
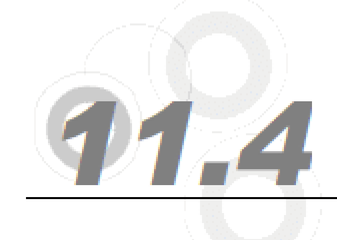


图 11-8　链接外部 CSS 样式后的显示效果



11.4 本章习题

一、选择题

1．下面说法错误的是(　)。

A. CSS 样式表可以将格式和结构分离

B. CSS 样式表可以控制页面的布局

C. CSS 样式表可以使许多网页同时更新

D. CSS 样式表不能制作体积更小下载更快的网页

2．CSS 样式表不可能实现(　)功能。

A. 将格式和结构分离　　B. 一个 CSS 文件控制多个网页

C. 控制图片的精确位置　　D. 兼容所有的浏览器

3．若要在网页中插入样式表 main.css，以下用法中，正确的是(　　)。

A. <Link href="main.css" type=text/css rel=stylesheet>

B. <Link Src="main.css" type=text/css rel=stylesheet>

C. <Link href="main.css" type=text/css>

D. <Include href="main.css" type=text/css rel=stylesheet>

二、填空题

1．CSS 又称为级联样式表，是 Cascading Style Sheet 的缩写，通常也简称为______。是 W3C 组织制定的，用于控制_________的一种标记性语言。

2．在页面中使用 CSS 样式，通常有 3 种方法：________________、______________和采用链接的形式调用。

三、简答题

1．什么是 CSS？

2．练习使用 3 种调入 CSS 的方法。

Chapter 12

CSS的语法

在 CSS 中，选择符的种类很多，每种选择符的优先级也不同。这些选择符包括，id 选择符、类选择符、类型选择符、伪类等。同时还可以使用子选择符对样式进行精确定位。或者使用选择符分组的方法，同时定义几个选择符中的样式。具体内容在本章中将一一讲述。

本章主要内容有:

◎ 熟悉 CSS 中的各类选择符的属性及用法。

◎ 了解 CSS 中属性和属性值的特点。

◎ 简单了解块元素和内联元素。

12.1 选择符

在 CSS 中，选择符的种类很多，每种选择符的优先级也不同。这些选择符包括，id 选择符、类选择符、类型选择符、伪类等。同时还可以使用子选择符对样式进行精确定位，或者使用选择符分组的方法，同时定义几个选择符中的样式。其具体内容如下所示。

12.1.1 id 选择符

id 选择符是所有选择符中优先级最高的。其语法结构如下所示。

```
#选择符名称
```

在定义 id 选择符的时候，必须使用英文的“#”加选择符的名称。在页面元素中，使用相应的 id 属性值，来调用 id 选择符中定义的 CSS 样式。

下面是一个使用 id 选择符的示例，其代码如下所示。

例程 12-1 id.html

```
01  <html >
02  <head>
03  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
04  <title>无标题文档</title>
05  <style>
06  #man
{
    width:450px;
    height:180px;
    background:#FF33CC;
}
07  </style>
08  </head>
09  <body>
    <div id="man">这里是一个使用 id 选择符的示例</div>
10  </body>
11  </html>
```

以上的代码中的 06 行代码就是 id 选择符。

在这个 id 选择符“#man”中，定义了宽度属性值为 450px，高度属性值为 180px，背景颜色为紫色。同时在主体元素中的<div>元素中，定义 id 属性值与 id 选择符“#”后面的内容相同。代码运行后，页面的显示效果如图 12-1 所示。

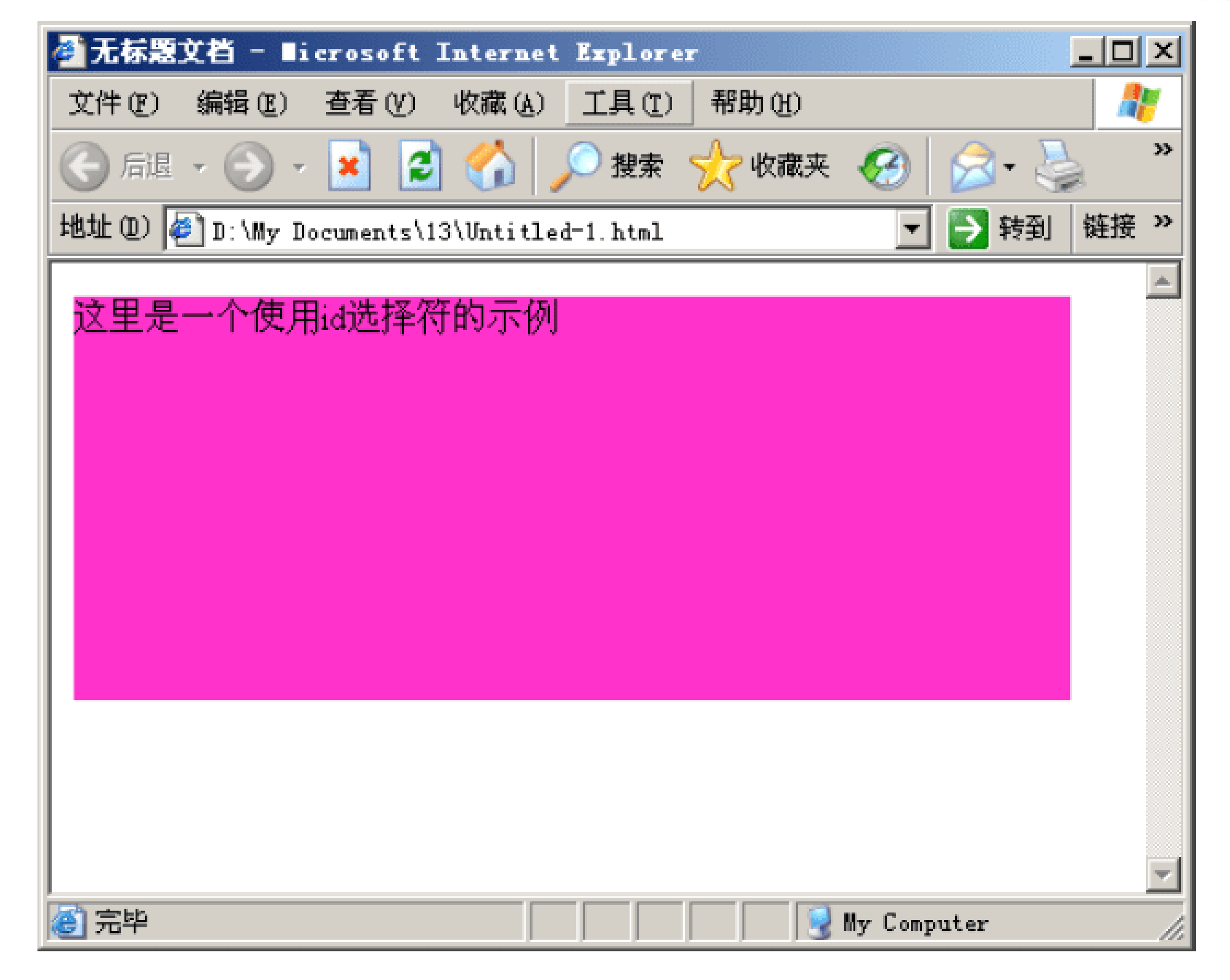


图 12-1　使用 id 选择符后的显示效果

12.1.2　类选择符

类选择符用来定义页面中可以重复使用的 CSS 属性，其优先级低于 id 选择符。其语法结构如下所示。

```
.选择符名称
```

在定义类选择符的时候，要使用英文的“.”加选择符的名称。在页面元素中，使用相应的 class 属性值，来调用类选择符中定义的 CSS 样式。

下面是一个使用类选择符的示例，其代码如下所示。

例程 12-2　example.html

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>类选择符</title>
<style>
.main
{
width:350px;
height:120px;
background:#999999;
}
</style>
</head>
<body>
<div class="main">这是一个使用类选择符的示例，其中的.main 就是我们定义的类选择符</div>
</body>
</html>
```

以上的代码中，06 行中在类选择符“.main”中，定义了宽度属性值为 350px，高度属性值为 120px，背景颜色为深灰色。同时在主体元素中的<div>元素中，定义 class 属性值与类选择符“.”后面的内容相同。代码运行后，页面的显示效果如图 12-2 所示。

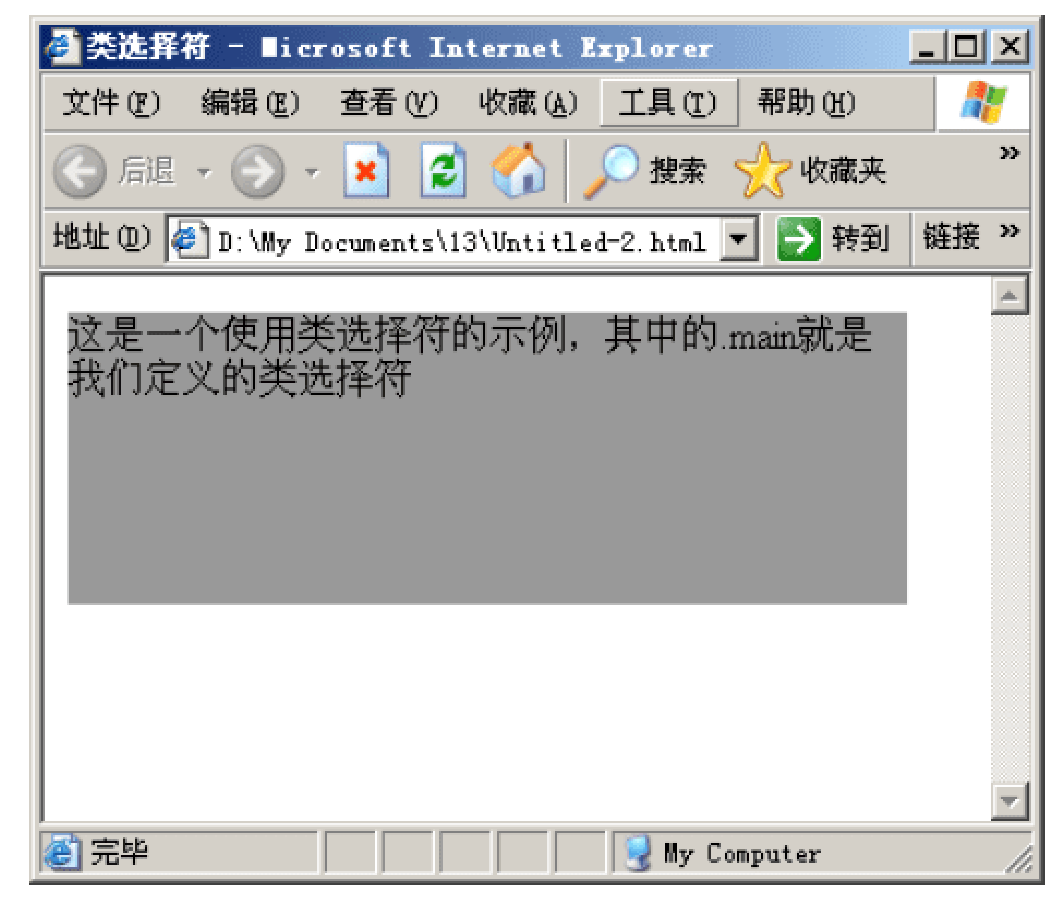


图 12-2　使用类选择符后的显示效果

12.1.3　类型选择符

类型选择符是按照 XHTML 中使用的相应元素名称来定义的 CSS 选择符。其语法结构如下所示。

```
选择符名称
```

在使用类型选择符的时候，可以直接使用元素的名称定义，在页面中所有与类型选择符相同的元素，都将使用该元素中定义的样式。

下面是一个使用类型选择符的示例，其代码如下所示。

例程 12-3　example.html

```
<html >
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>类型选择符</title>
<style>
p{
width:300px;
height:50px;
color:#ffffff;
background:#666666;
border:4px solid #000000;
padding:6px;}
</style>
</head>
<body>
```

```
<div>使用 CSS 的类型选择符</div>
<p>这里就是类型选择符作用的地方</p>
<p><img src="pic.jpg" width="200" /></p>
10  </body>
11  </html>
```

以上的代码中，06 行在类型选择符 p 中，定义了宽度属性值为 300px，高度属性值为 50px，背景颜色为深灰色。然后在页面中定义了一个<p>元素。代码运行后，页面的显示效果如图 12-3 所示。



图 12-3　使用类型选择符后的显示效果

从图 12-3 可用看出，当使用类型选择符后，使用相同名称的相关元素的样式发生了改变。但是因为图片的关系，在第二个类型选择符中高度适应了图片的高度。

12.1.4　伪类

伪类指的是一类特殊的选择符，通过这类选择符，可以定义某种鼠标触发时间的显示效果。其语法结构如下所示。

```
选择符名称:伪类名称
```

在使用伪类的时候，要对应某个其他的选择符，这个选择符可以使用 id 选择符、类选择符或者类型选择符，但通常会将伪类定义在链接内容。伪类都有固定的写法，一般格式为英文的“:”加伪类的名称。

下面是一个使用伪类选择符的示例，其代码如下所示。

例程 12-4　example.html

```
<html >
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>伪类</title>
<style>
a:hover
  {
    display:block;
    width:400px;
    height:100px;
    background:#cccccc;
    border:4px solid #000000;
    padding:6px;
  }
</style>
```

```
08    </head>
09    <body>
      <a href="#">这是一个使伪类的示例</a>
10    </body>
11    </html>
```

以上的代码中，06 行在 a 选择符后面，定义了“:hover”伪类。这个伪类的含义是：定义鼠标悬停时候的显示效果。在伪类选择符中，首先定义了显示方式属性(display)的取值为块属性(block)。然后定义了宽度属性值为 400px，高度属性值为 100px，背景颜色为浅灰色，并加了宽度为 4 的黑色边框。代码运行后，当鼠标悬停在链接内容上时，页面的显示效果如图 12-4 所示。

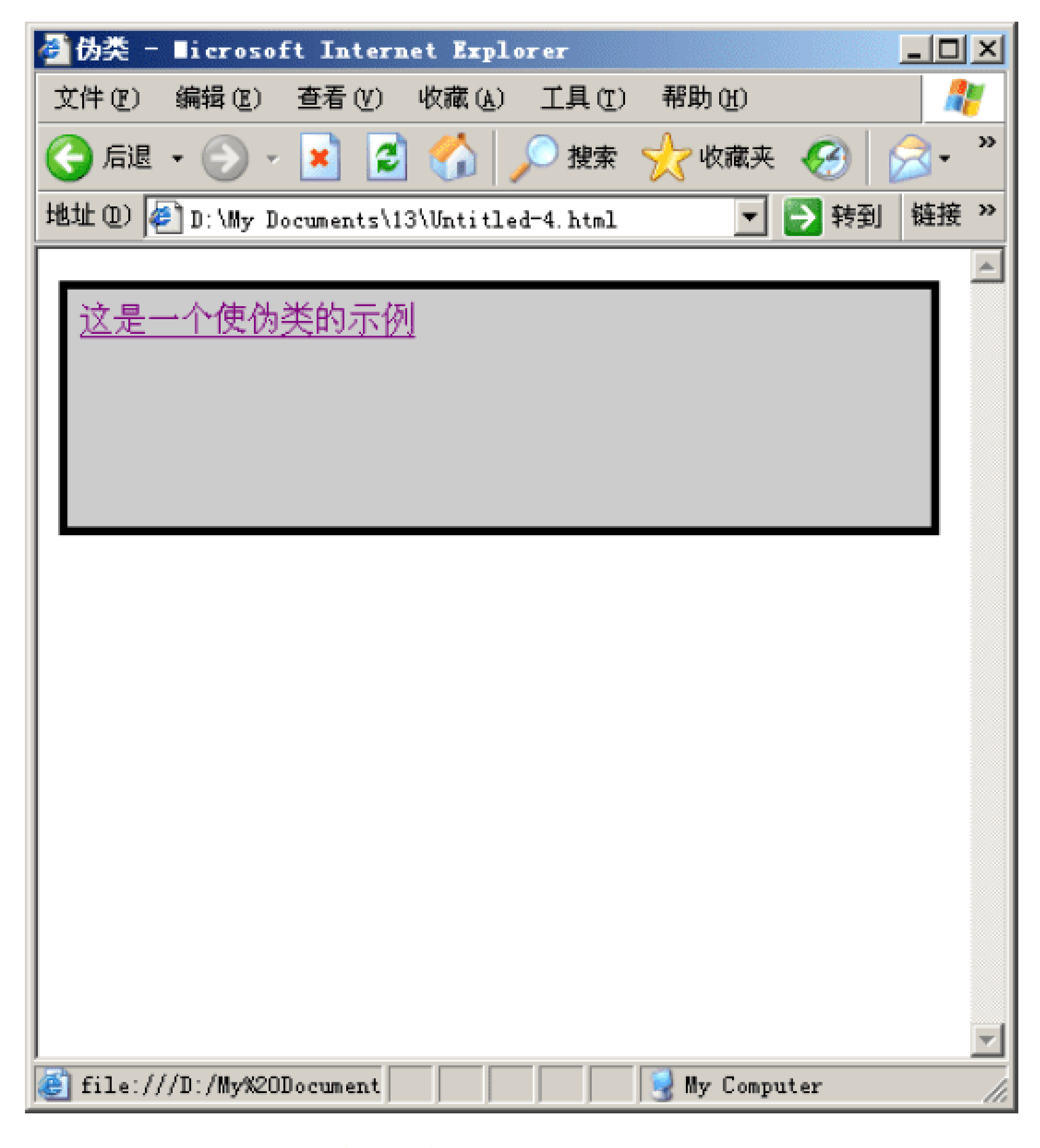


图 12-4　使用伪类选择符后的显示效果

12.1.5　子选择符

子选择符用来精确定位某个元素。其语法结构如下所示。

```
选择符 1  选择符 2
```

在每个选择符之间使用空格进行分隔，其含义为：在第一个选择符中使用第二个选择符的内容。

注意

在使用子选择符的时候，选择符所在的元素之间要有嵌套关系，否则子选择符不能够正常发挥作用。

下面是一个使用子选择符的示例，其代码如下所示。

例程 12-5 example.html

```
01  <html >
02  <head>
03  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
04  <title>子选择符</title>
05  <style>
06  div p
  {
    width:300px;
    height:100px;
    background:#cccccc;
    border:4px solid #000000;
  }
07  </style>
08  </head>
09  <body>
    <div>
    <p>这里是包含在 div 里的 p 元素,和 div 一起才有作用</p></div>
    <p>这里是独立的 p 元素</p>
10  </body>
11  </html>
```

以上的代码中，06 行使用子选择符，定义了元素的宽度属性值为 300px，高度属性值为 100px，背景颜色为浅灰色，并加了一个黑色边框。此时在<div>元素中的<p>元素，将使用子选择符中定义的样式。不具有子选择符所定义的包含关系的元素，无法使用子选择符中定义的样式。代码运行后，页面的显示效果如图 12-5 所示。

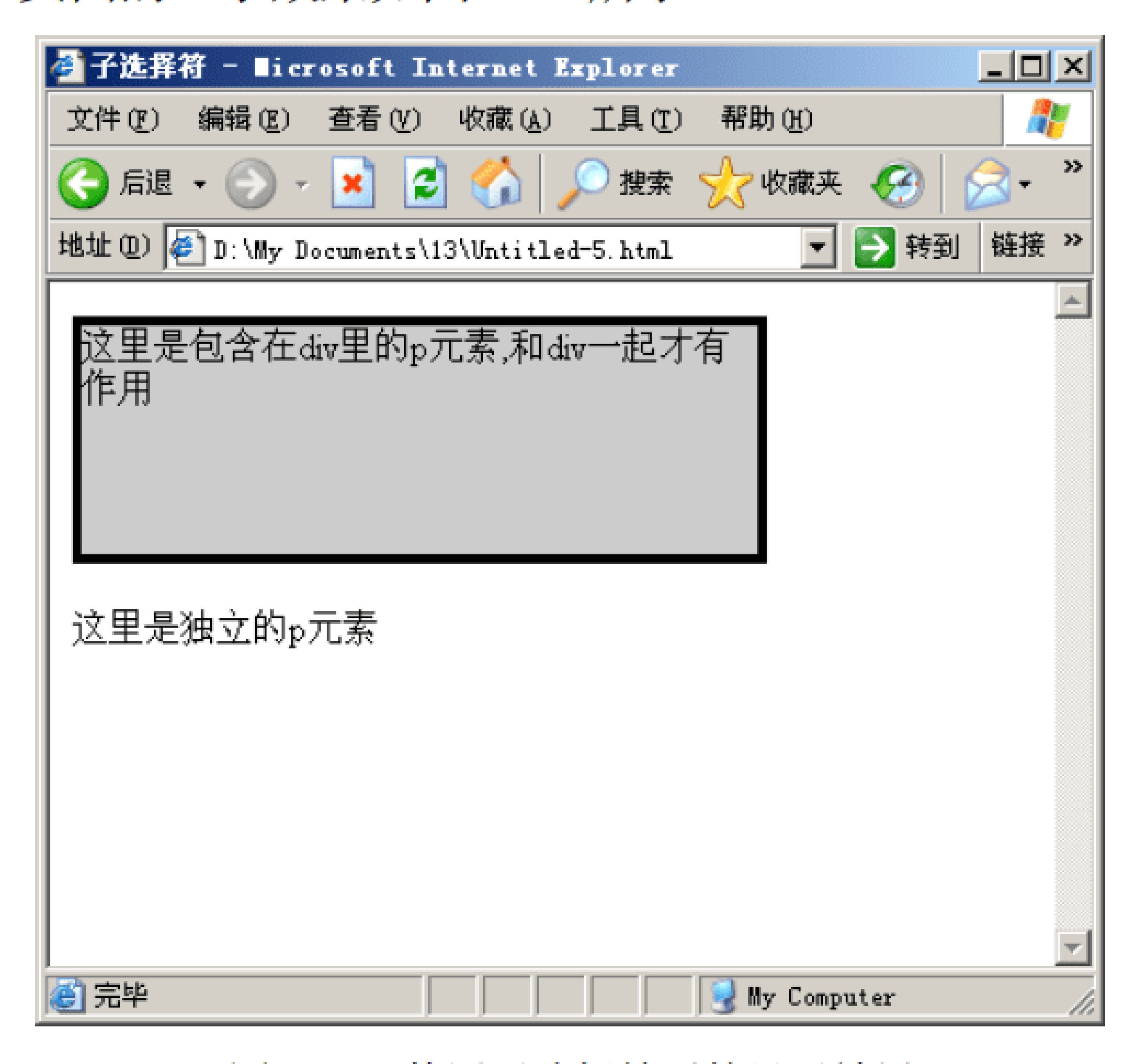


图 12-5　使用子选择符后的显示效果

12.1.6 选择符分组

使用选择符分组的方法，可以一次性定义几个选择符中的样式。其语法结构如下所示。

```
选择符 1,选择符 2
```

在使用选择符分组的时候，每个选择符之间，使用英文的逗号分隔。下面是一个使用选择符分组的示例，其代码如下所示。

例程 12-6 example.html

```
01  <html>
02  <head>
03  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
04  <title>无标题文档</title>
05  <style>
06  div,p,h1
  {
    width:300px;
    height:100px;
    background:#cccccc;
    border:4px solid #009933;
  }
07  </style>
08  </head>
09  <body>
    <div>这里是一个独立的 div 元素</div>
    <p>这里是独立的 p 元素</p>
    <h1>这里是独立的 h1 元素</h1>
10  </body>
11  </html>
```

以上的代码中，06 行使用选择符分组的方法，统一定义了<p>元素、<div>元素、<h1>元素的宽度属性值为 300px，高度属性值为 100px，背景颜色为浅灰色，并有一个绿色的边框。代码运行后，页面的显示效果如图 12-6 所示。

使用选择符分组的方法，可以方便地一次性定义几个选择符中相同得属性，提高了 CSS 样式的定义的效率。

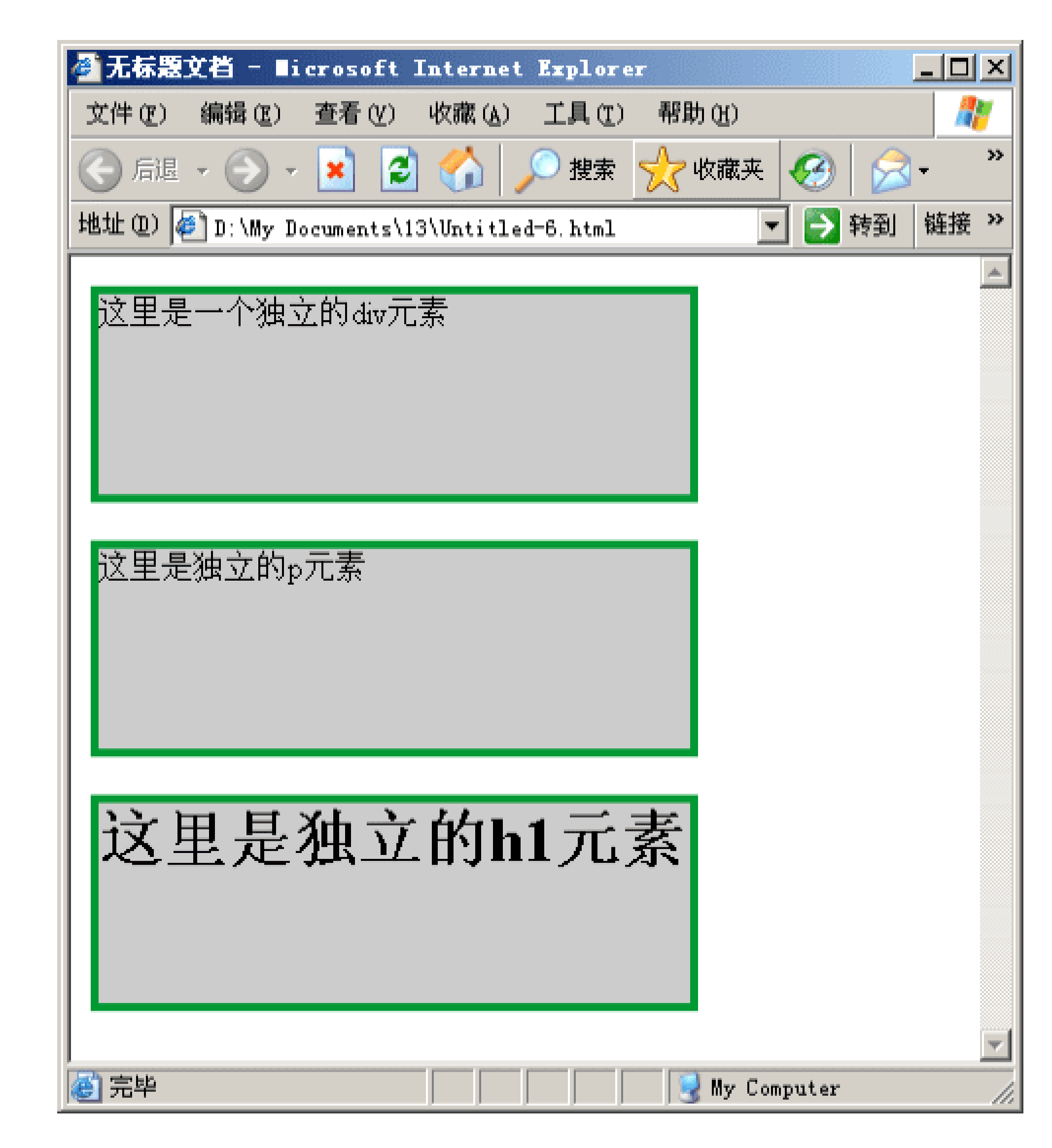


图 12-6 使用选择符分组定义 CSS 属性后的显示效果

12.1.7 选择符的优先级

在各种选择符中，id 选择符的优先级最高，然后是类选择符，最后是类型选择符。而

伪类选择符必须定义在相应的选择符之后，所以伪类选择符的优先级和伪类前面的选择符一致。

下面是一个关于选择符优先级的示例，其代码如下所示。

例程 12-7 example.html

```
01  <html>
02  <head>
03  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
04  <title>选择符的优先级</title>
05  <style>
06  p
  {
    width:400px;
    height:100px;
    background:#cccccc;
  }
07  .p
{
    width:300px;
    background:#666666;
}
08  #p
  {
    width:300px;
    height:100px;
    background:#999999;
    border:4px solid #009933;
  }
09  </style>
10  </head>
11  <body>
    <p>三种选择符的优先级</p>
    <p class="p">三种选择符的优先级</p>
    <p class="p" id="p">三种选择符的优先级</p>
12  </body>
13  </html>
```

以上的代码中，06、07 和 08 行分别定义了 3 个选择符。在类型选择符 p 中，定义宽度属性值为 400px，高度属性值为 100px，背景颜色为浅灰色。在类选择符“.p”中，定义宽度属性为 300px，背景颜色为深灰色。在 id 选择符“#p”中，定义宽度属性值为 300px，高度属性值为 100px，背景颜色为灰色。在第 1 个<p>元素中，未定义任何属性。在第 2 个<p>元素中，定义了类属性值为“p”。在第 3 个<p>元素中，同时定义了类属性值为“p”，id 属性值也为“p”。代码运行后，页面的显示效果如图 12-7 所示。

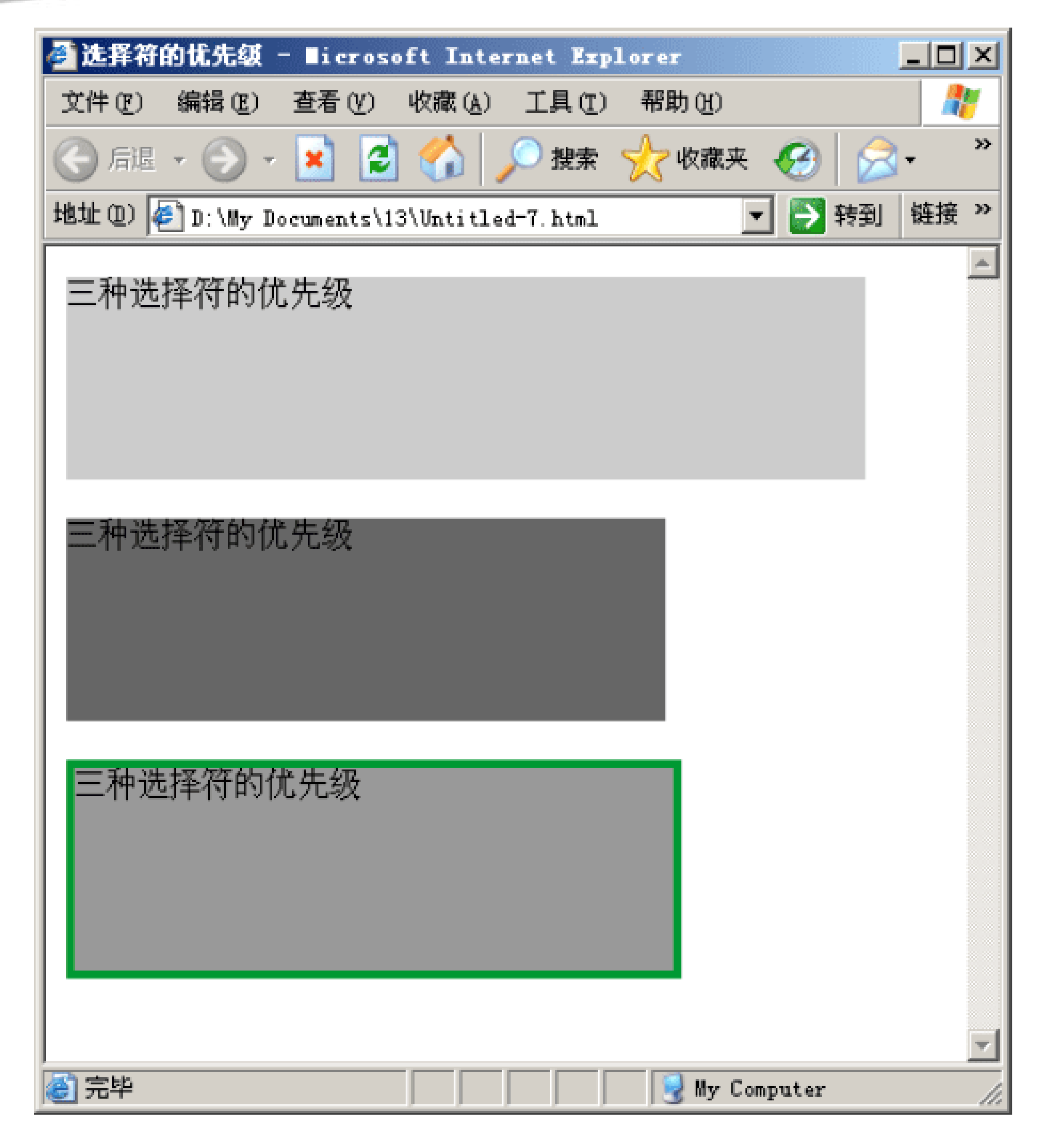


图 12-7　选择符优先级的显示效果

从图 12-7 可以看出，当同时定义类型选择符和类选择符时，类选择符中定义的属性会覆盖类型选择符中定义的相同的属性。在同时使用 id 选择符和类选择符的时候，id 选择符中定义的属性会覆盖类选择符中定义的相同的属性。没有被覆盖的属性会依然作用在元素上。

12.2 属性

在 CSS 中，属性是控制各种页面效果的核心。所有页面效果的实现都要通过定义相应的属性来定义。在每个属性中，定义的值可以各不相同，每个属性值都有固定的写法。全面地掌握各种 CSS 属性，可以更方便地控制页面的表现形式。

在样式表中，属性的作用和元素中属性的作用基本相同，用来定义元素(或内容)的显示效果，包括文本属性、链接属性、字体属性等很多内容。

在样式表中，属性的名称都是固定的。关于样式表中都有哪些属性，每种属性的显示效果等知识，将在后面的章节中详细讲解。下面是一个使用属性的实例。其代码如下所示。

例程 12-8　example.html

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>CSS 的属性</title>
<style>
p{
font-size:26px;
```

```
        color:#333333;
        background:#FF00FF;
        border:5px solid #000000;}
07  </style>
08  </head>
09  <body>
    <p >这里就用了几个 CSS 的属性</p>
10  </body>
11  </html>
```

在该实例中，06 行中类型选择符 p 中的“font-size”、“color”等就是样式中使用的属性。其代码运行后，显示效果如图 12-8 所示。

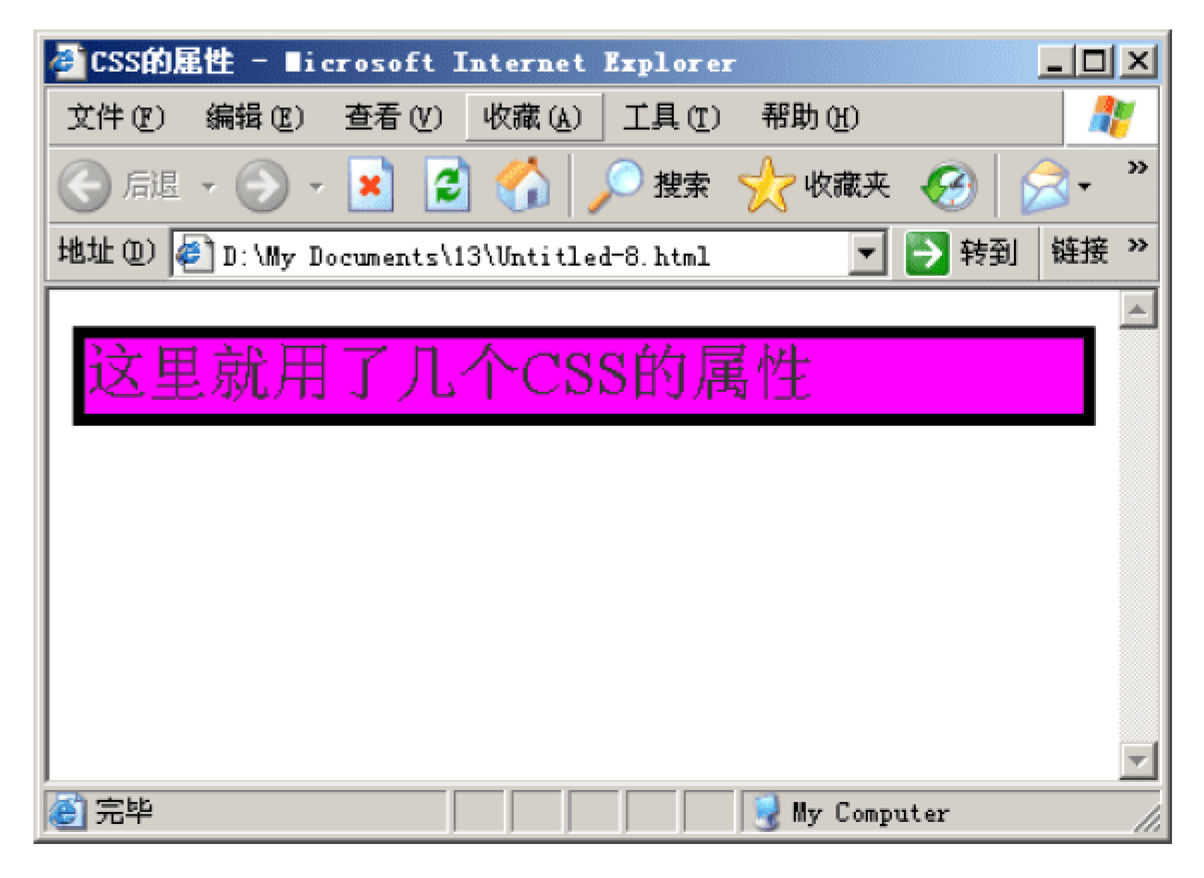


图 12-8 在样式表中使用属性的显示效果

12.3 值

在样式表中，每个属性都有相应的值与之对应，值用来确定属性的最终效果。在属性对应的值中，有一些是通常使用的值，如长度值、颜色值等。还有一些是某个属性所特有的值。每种取值的具体写法，将在后面的章节中作详细讲解。下面是一个使用值的实例。其代码如下所示。

例程 12-9 example.html

```
01  <html>
02  <head>
03  <meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
04  <title>CSS 实例页面</title>
05  <style>
06  p{
    font-size:26px;
    color:#333333;
     background:#FF00FF;
```

```
        border:5px solid #000000;
07      </style>
08      </head>
09      <body>
10      <p>使用 CSS 的实例</p>
11      </body>
12      </html>
```

在该实例中，06 行中类型选择符 p 中“font-size”、“color”“background”、“border”后面定义的 26px、 #333333、#FF00FF、5pxsolid #000000 等就是属性相对应的值。它们和属性一起作用，其代码运行后，显示效果如图 12-8 所示。

12.3.1 颜色值

在 CSS 中，定义颜色的方法有很多种，其中比较常用的方法有使用颜色名称和使用十六进制颜色值两种。

1．使用颜色名称

使用颜色名称可以定义简单的颜色效果。只有一定数量的颜色名称可以被浏览器支持。下面是一个使用颜色名称来定义颜色的示例，具体代码如下所示：

```
p {color:red}
```

该样式定义了段落中文本颜色为红色。通常主要的一些浏览器都能识别的颜色名称和中文翻译如下所示：

red(红)、yellow(黄)、blue(蓝)、silver(银)、teal(深青)、white(白)、navy(深蓝)、olive(橄榄)、purple(紫)、gray(灰)、green(绿)、lime(浅绿)、maroon(褐)、aqua(水绿)、black(黑)、fuchsia(紫红)

2．使用十六进制颜色

使用十六进制颜色可以设置较为复杂的颜色，通常在制作网页的时候，各种颜色的设置都使用十六进制的颜色值。下面是一个用十六进制设置颜色的示例，其具体代码如下所示：

```
p {color:#ff6633;}
```

在使用十六进制颜色时一定要在颜色值前加“#”。六位数字中，前两位代表红色的值，中间两位代表绿色的值，最后两位代表蓝色的值。每组色值的数字越大则代表含有的成分越多。比如#000000 代表没有任何三原色即为黑色。#ffffff 是三原色均为最大值也就是白色。

因为相同的颜色在不同的操作系统或者浏览器上可能会显示不同的效果，而有一些颜色在各种情况下都能显示基本相同的颜色，这些样色就被称为网络安全色。网络安全色，用十六进制表示，分别是相应的红、绿、蓝色值为 00、33、66、99、cc、ff 的组合。比如“#ff0000”就是一种网络安全色，也就是颜色名称中的“red”。组合后的网络安全色有 216 种。在前面的例程中已经用过这些颜色值，这里就不再举实例了。

12.3.2 长度值

在 CSS 中，长度单位分为绝对长度单位和相对长度单位。绝对长度单位，是指在各种环境中都使用相同的长度值。相对单位是指：根据各自定义的标准和环境的不同，显示的长度会发生改变。下面进行详细讲解。

1. 绝对长度单位

绝对长度单位一般为常用的各种长度单位值，具体包括 in(英寸)、cm(厘米)、mm(毫米)、pt(磅)、pc(pica)。下面对其中相对不常见的属性值进行相应的说明。

- pt(磅)：标准印刷上的单位。72 个磅的长度为一英寸。
- pc(pica)：印刷上用的单位。1pc 的长度是 12 磅。

绝对长度一般在打印文档定义的样式中比较常用，在网页的设计中很少使用。

2. 相对长度单位

相对长度单位是网页中常用的长度单位值，具体包括 em、ex、px。下面进行详细讲解。

- em：定义文字使用 font-size 属性定义的值。例如，在 font-size 属性中，定义文字大小为 12pt，那么此时 1em 就是 12pt 的长度。
- ex：定义文字使用字母 x 的高度。因为不同的字体中 x 的高度是不同的，所以单位 ex 的实际大小也不同。
- px(像素)：将显示器分成非常细小的方格，那么每个方格就是一个 px。px 的实际显示大小要受到显示器分辨率的影响。显示器的分辨率定义了显示器划分像素的方式。例如，将浏览器划分为 4 个部分，则 1 像素代表浏览器 1/4 的大小。如果将浏览器分为 9 个部分，那么 1 像素就代表浏览器 1/9 的大小。

12.3.3 百分比值

百分比值是指：通过一个指定的标准，定义需要显示的长度。根据指定的标准不同，最终显示的长度会有所区别。百分比值的写法如下所示。

```
数字%
```

在百分比值中，数字的取值可正可负。例如，定义一个元素的宽度为 100px，那么如果定义其中所包含的一个元素的宽度为 20%，那么这个元素的宽度就是 20 像素。下面是一个使用百分比值的实例。其代码如下所示。

例程 12-10 example.html

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>CSS 实例页面</title>
<style>
```

```
06  div{
    width:450px;
    height:200px;
    border:5px solid #333333;}
07  p{
    width:60%;
    height:50%;
    font-size:16px;
    background-color:yellow;
    color:block;
    border:4px solid #000000;
    margin:10px;}
08  </style>
09  </head>
10  <body>
11  <div>
    <p>使用 CSS 的实例</p>
12  </div>
13  </body>
14  </html>
```

该实例中，06 行使用长度值定义了<div>元素的宽度和高度，同时 07 行使用百分比值定义了<p>元素的宽度和高度。其代码运行后，显示效果如图 12-9 所示。

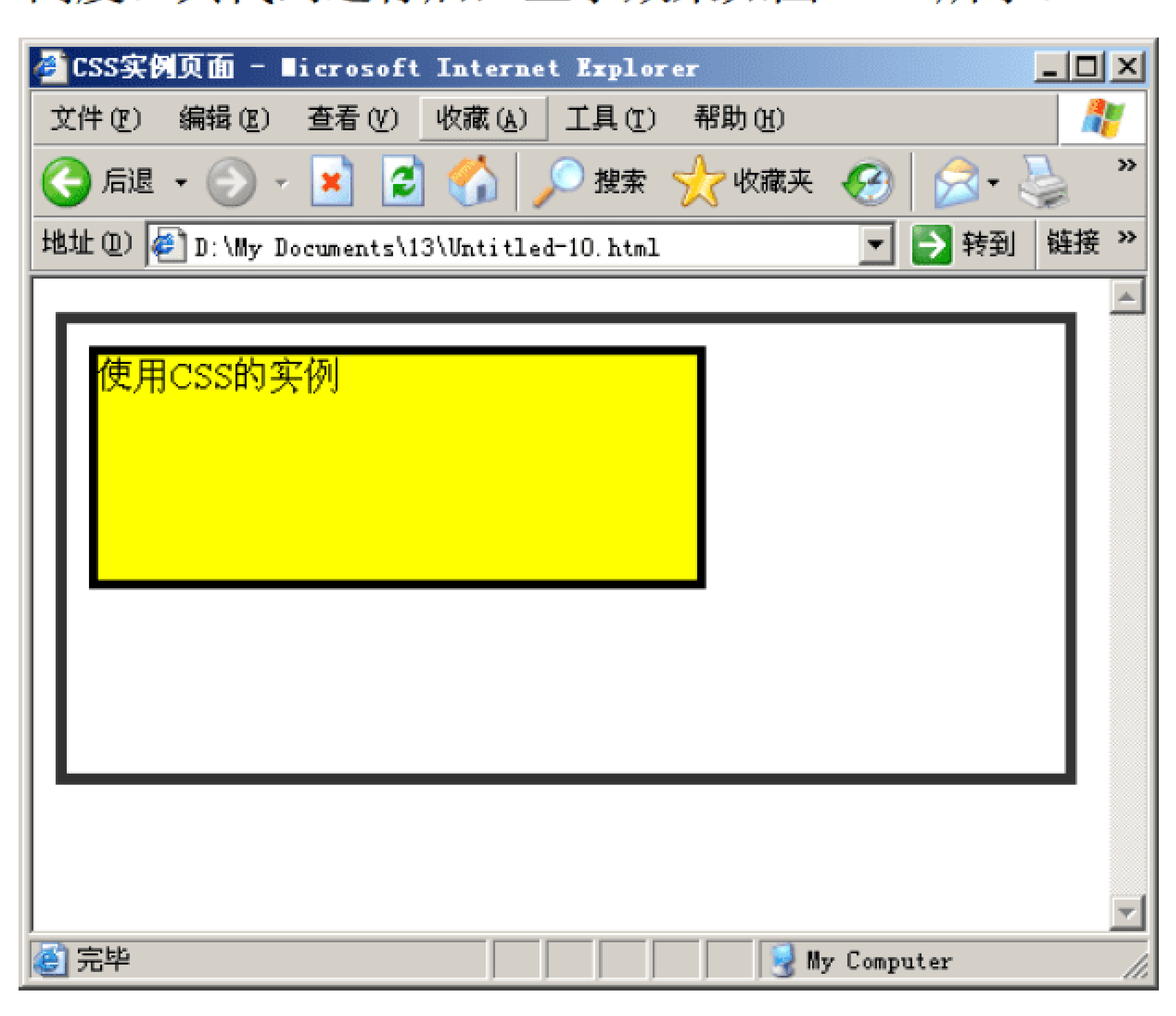


图 12-9　样式表中使用百分比值的显示效果

12.4 继承值

继承值是指：在 CSS 中当在某个元素中定义了某个属性值，其包含的各种元素都会使用这个属性值。不是所有的 CSS 属性都具有继承性。合理地使用 CSS 的继承性，可以更方便地

对各种效果进行控制。

下面是一个使用继承性的实例，其代码如下所示。

例程 12-11 example.html

```
01  <html >
02  <head>
03  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
04  <title>继承值</title>
05  <style>
06  .main
{
    color:red;
}
07  </style>
08  </head>
09  <body>
    <div class="main">这里直接引用样式，显示的是直接定义颜色的部分
    <div>这里显示的是继承颜色的部分。</div></div>
10  </body>
11  </html>
```

以上的代码中，06 行首先定义了类属性值为 main 的元素中，文本的颜色为砖红色。然后在元素及其内部的<div>元素中定义了部分文本内容。代码运行后的显示效果如图 12-10 所示。

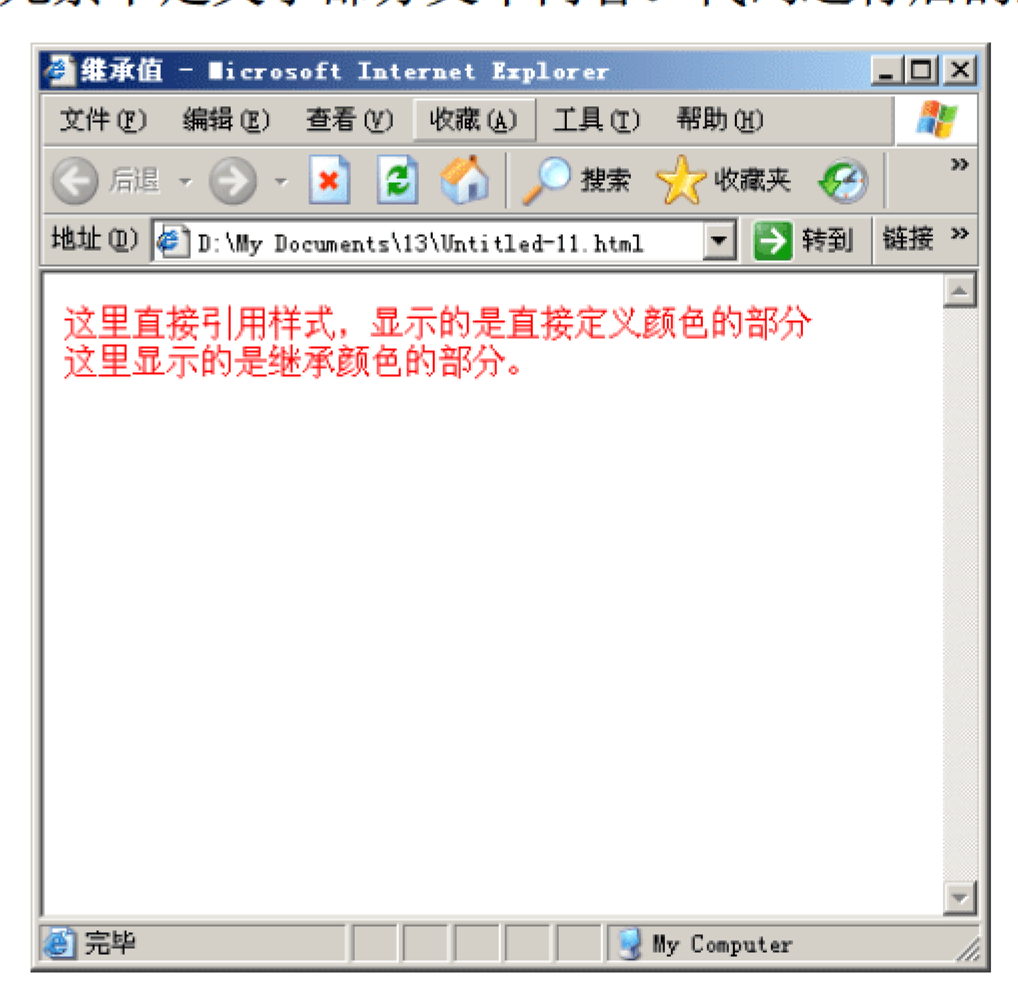


图 12-10　继承值的显示效果

从图 12-10 可以看出代码中独立使用的<div>元素中的内容在页面中也为砖红色。

12.5 默认值

默认值是指：在 CSS 中没有定义任何值的时候，元素所使用的值。大多数的默认值都是 none 或者为 0，部分属性中有自身特有的默认值，如 left、auto 等。但是在不同的浏览器中某

些属性的默认值会有变化。合理地使用属性的默认值可以减少很多的代码。

下面是一个使用默认值的实例，其代码如下所示。

例程 12-12 css-valuepercent.html

```
01  <html >
02  <head>
03  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
04  <title>默认值</title>
05  <style>
06  .main
{
    width:400px;
    height:200px;
    }
07  div
  {
    border:2px solid #000000;
  }
08  </style>
09  </head>
10  <body>
    <div class="main">
    <div>这是一个使用默认值的实例。</div></div>
11  </body>
12  </html>
```

以上的代码中，06 行首先定义了类属性值为 main 的元素的宽度为 400px，高度为 200px。然后定义页面中<div>元素的边框为 2 像素黑色实线边框。在内部的<div>元素中，定义了部分文本内容。代码运行后的显示效果如图 12-11 所示。

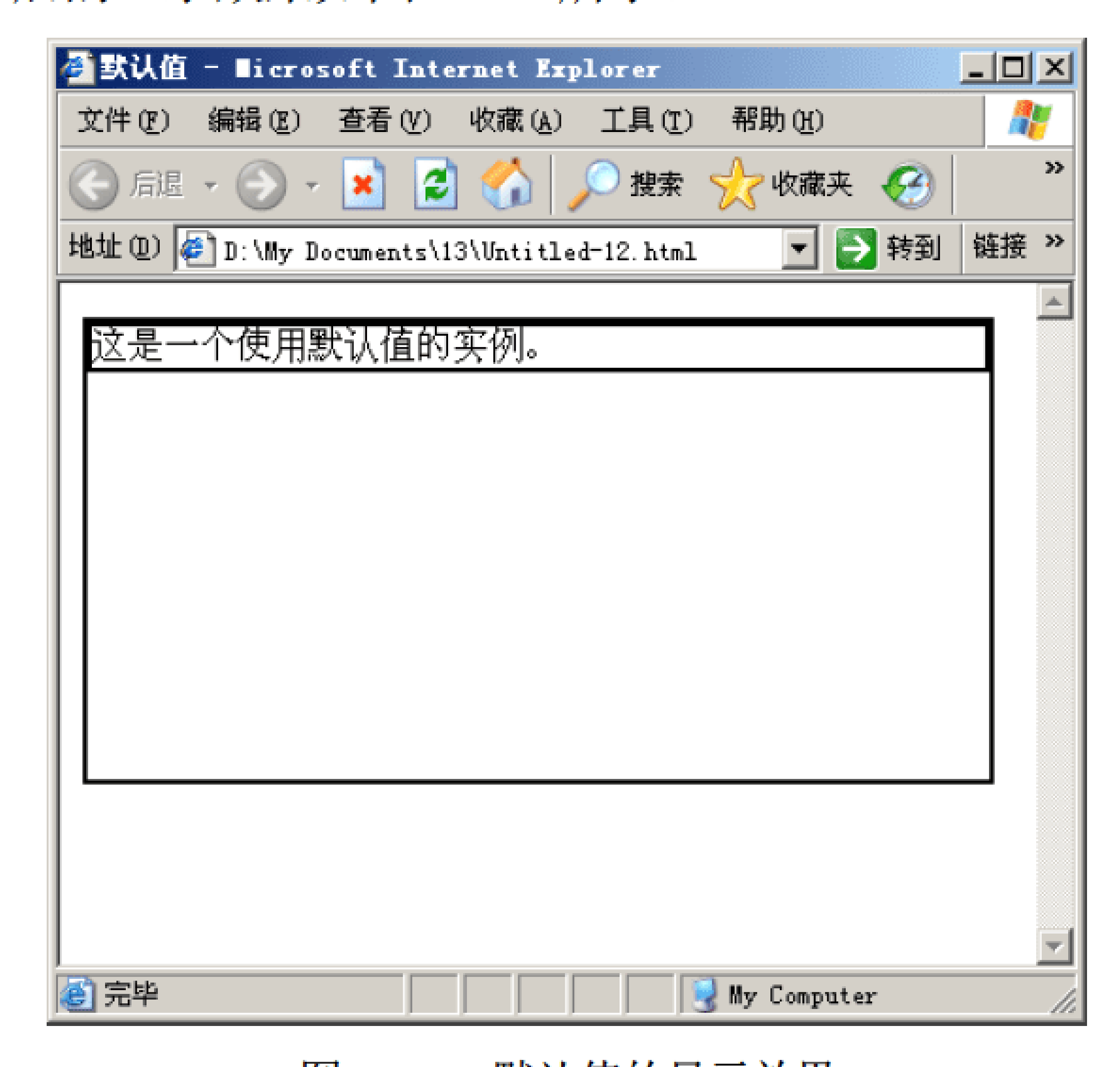


图 12-11 默认值的显示效果

从图 12-11 可以看出，虽然在内部的<div>元素中未定义任何宽度属性，但是此时<div>元素的宽度依然和主体元素的宽度相同。这是因为<div>元素的默认值为 auto(即随着主体元素的宽度的变化，自动缩放显示宽度)，所以此时子元素会以宽度 100%显示。

12.6 块元素和内联元素

在 HTML 中，页面元素可以分为两类。一类是块元素，另一类是内联元素。本节将分别讲解它们的各自特点，注意这两个元素的区别和联系。

12.6.1 块元素

块元素是指那些在默认的情况下会换行显示的元素，如<div>元素、<p>元素等。之所以要将元素区分为块元素，是因为在样式表中，有部分属性是只对块元素起作用的。下面是一个使用块元素的实例。其代码如下所示。

例程 12-13 example.html

```
01  <html>
02  <head>
03  <meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
04  <title>使用块元素</title>
05  <style>
06  div{
    border:5px solid red;}
07  p{
    border:5px solid yellow;}
08  </style>
09  </head>
10  <body>
    <div>这里是使用块元素区域中的内容</div>
    块元素区域以外的内容
    <p>这里是使用块元素区域中的内容</p>块元素区域以外的内容
11  </body>
12  </html>
```

在该实例中，为了使块元素显示得更加明显，06 和 07 行使用了样式表定义的元素的边框。其代码运行后，显示效果如图 12-12 所示。

从图 12-12 可以看出，块元素在默认的情况下，在水平方向上不允许任何元素与其同行显示。

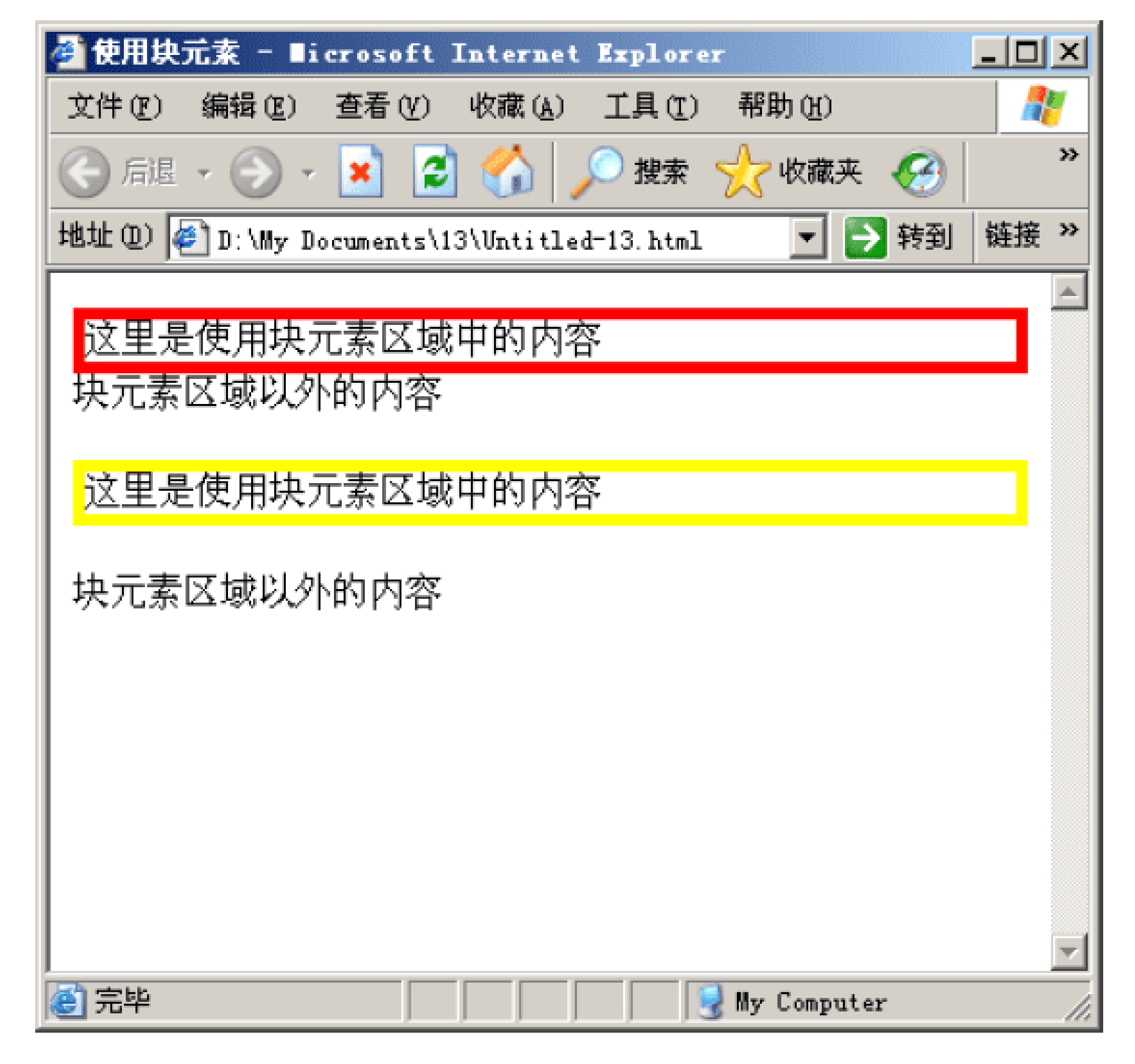


图 12-12 块元素的显示效果 1

12.6.2 内联元素

内联元素是指在默认的情况下，显示方式类似于文本的元素。内联元素默认情况下同行显示，直到总体的宽度大于父元素的宽度时，才换行显示。下面是一个使用内联元素的实例。其代码如下所示。

例程 12-14 example.html

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>内联元素</title>
</head>
<body>
<strong>这是一个使用内联元素的示例</strong>
这是一张图片：
<img src="pic.jpg" width="200" />
这里是一个按钮：
<input type="button" value="下一步" name="name1" />
</body>
</html>
```

在代码中没有定义任何的样式，整个程序都是默认显示。其代码运行后，显示效果如图 12-13 所示。



图 12-13　内联元素的显示效果

从图 12-13 中可以看出，因为第一句话过长，与图片一起时超过了页面宽度，因而图片换行显示，而图片、第三句话和按钮没超过页面宽度，在默认的情况下，它们同行显示。

12.7 应用样式的优先级

通过前面的学习，可以了解到，在页面中使用 CSS 的方法有 3 种，分别为在元素中直接定义、在页面头部调用样式、从外部文件调用样式。在这 3 种使用 CSS 样式的方法中，在元素中直接定义 CSS 的优先级最高，其次是在页面头部调用的 CSS，最后是从外部文件中调用的 CSS 样式。

下面是一个关于应用样式优先级的实例，其 HTML 页面中的代码如下所示。

例程 12-15 css-valuepercent.html

```
<html >
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>无标题文档</title>
<link href="style/main.css" rel="stylesheet" type="text/css" />
<style>
div
  {
  background:#666666;
  color:#ffffff;
  }
</style>
```

```
</head>
<body>
<div style="background:#000000;">这是关于应用样式优先级的实例</div>
</body>
</html>
```

在<linik>元素中，被调用页面 main.css 中的代码如下所示：

```
div
{
  background:#ffffff;
  color:#333333;
  width:500px;
  height:100px;
}
```

在以上的代码中，使用元素中定义 CSS 的方法，定义<div>元素的背景颜色为黑色。在头部的<style>元素中，定义背景颜色为灰色，文本颜色为白色。在外部的 CSS 文件中，定义背景颜色为白色，文本颜色为深灰色，宽度为 500px，高度为 100px。页面最终的显示效果如图 12-14 所示。

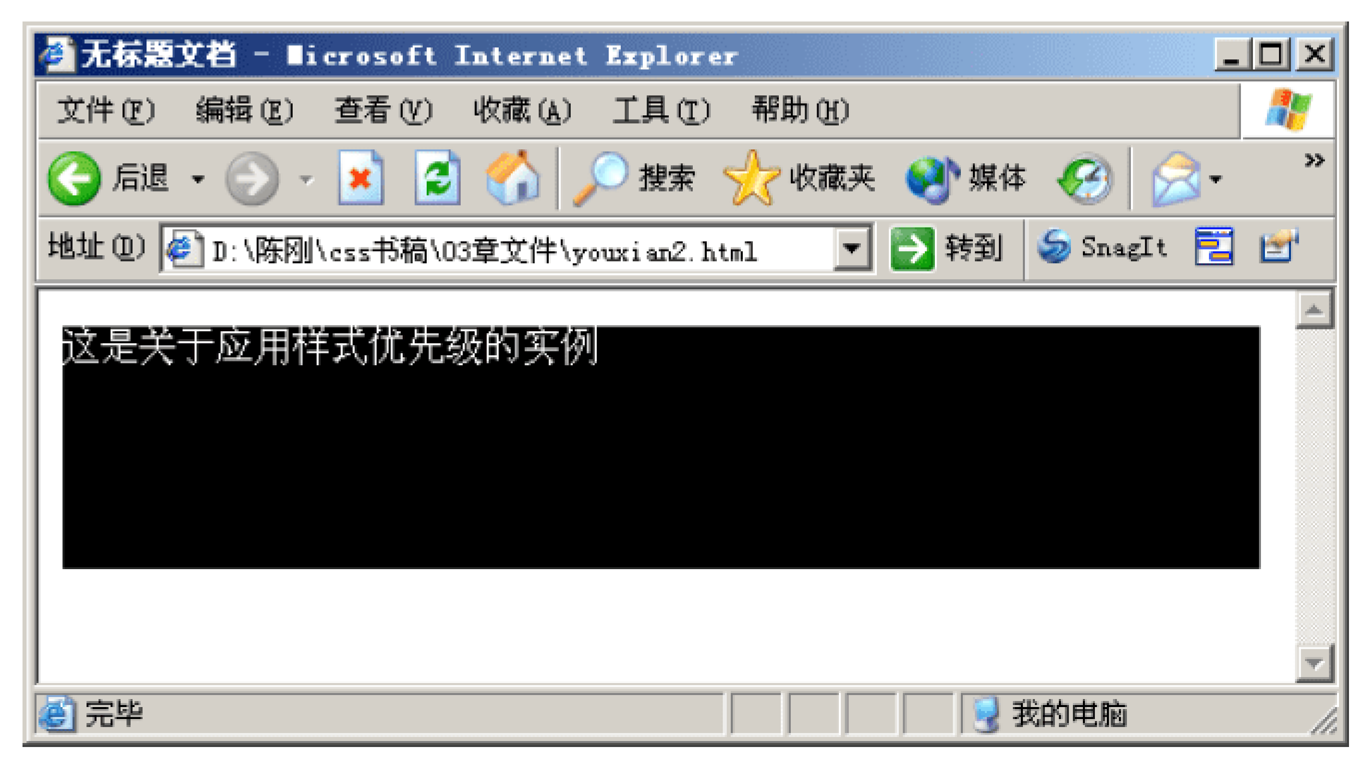


图 12-14　3 种方式调用样式表的优先级

从图 12-14 可以看出，页面最终在定义的 3 个背景颜色中，使用了最优先的元素内部定义的颜色。文本的颜色使用页面头部样式中定义的颜色。而宽度和高度则使用外部样式中定义的大小。在 3 种使用样式的方法中，元素中直接使用样式表的方法的优先级最高，然后是从页面头部调用的样式表，最后是从外部调用的样式表。下面是一个在元素中使用不同方法应用样式的实例。其代码如下所示。

例程 12-16　css -important.html

```
<html >
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>无标题文档</title>
<link href="style.css" rel="stylesheet" type="text/css" />
<style>
```

```
h3{
font-size:36px;}
img{
border:20px solid #999999;}
</style>
</head>
<body>
<h3 style="font-size:24px;">CSS 演示实例</h3>
<p style="font-size:36px;">给图片加了边框</p>
<img src="pic.jpg"/>
</body>
</html>
```

调用的 style.css 文件中的代码如下所示。

```
h3{
    font-size:12px;
    font-weight:bold;}
img{
                                    border:5px solid #000000;}
```

其代码运行后，显示效果如图 12-15 所示。



图 12-15 不同方法应用样式的显示效果

从图 12-15 可以看出，此时元素<h3>和<p>中，字体使用的都是各自元素中 style 属性中直接定义的字体大小。<img>元素优先使用了<style>元素中定义的边框属性。

12.8 本章习题

一、选择题

1．各类选择符中，优先级最高的是(　　)。

A. id 选择符　　B. 类选择符　　C. 伪类　　D.子选择符

2．以长度单位中，属于绝对长度的是(　　)。

A.px　　B.ex　　C.em　　D.pt

3．以下选项中不能用来表示 CSS 颜色的是(　　)。

A. red　　B. #FF0000　　C. rgb(f, 0, 0)　D. rgb(100%, 0,0)

二、填空题

1．在 CSS 中，选择符的种类很多，这些选择符包括：id 选择符、__________、类型选择符、_________等。每种选择符的优先级也不同，其中________优先级最高。

2．继承值是指，在 CSS 中当在某个元素中定义了某个属性值，其包含的各种元素都会使用这个属性值。所以使用继承值时，必须先定义_________。不是所有的 CSS 属性都具有继承性。

3．长度单位分为___________和___________。

三、实战练习

编写一个综合应用各类选择符的程序，熟悉优先级的意义。

Chapter 13

CSS控制文本的显示

CSS 中，文本的控制包括两个方面的内容，一方面控制文本中字体的各种显示效果(如控制字体的大小等)，另一方面控制文本的显示效果(如文本的缩进等)。在 CSS 中文本的控制是很重要的内容，文本的显示效果直接影响读者对页面信息的读取。

本章主要内容有:

◎ 重点掌握 CSS 控制字体的显示方法并能熟练应用

◎ 了解 CSS 控制文本的显示方法及其使用方法

◎ 能熟练使用字体综合属性



13.1 控制字体的显示

在 CSS 中，字体的控制有控制文本的字体、字体的大小、字体的样式、字体的颜色、字体的修饰等方面的内容。本节将分别进行详细讲解。

13.1.1 字体选择属性 font-family

字体选择属性(font-family)用来定义文本中使用的那种字体。字体选择属性的值为字体名称，其语法结构如下所示。

```
font-family : name;
```



注意

在使用字体选择属性的时候，如果字体中含有空格，或者使用的是中文字体，要在字体上使用引号。如果同时定义了几种字体，则每种字体之间使用逗号分隔。最先定义的字体属性值会优先使用。

下面是一个使用字体选择属性的实例，其代码如下所示。

例程 13-1 font-family.html

```
<html >
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>字体的选择</title>
<style>
div
  {
    font-family:"隶书","黑体";
    width:350px;
    height:120px;
    background:yellow;
}
p
  {
    font-family:Arial, Helvetica, sans-serif;
    width:300px;
    height:120px;
    background:#999999;
  }
</style>
</head>
<body>
```

```
11  <div>这是使用中文字体的元素</div>
    <p>how do you do!</p>
12  </body>
13  </html>
```

以上的代码中，06 行定义<div>元素的字体属性值为隶书和黑体，隶书是一种常见的字体，所以<div>元素中会使用隶书。07 行在<p>元素中，定义字体属性值为 3 种英文字体。由于 Arial 是常用的英文字体，一般电脑中都会安装，所以在<p>元素中，最终会使用 Arial 字体。其显示效果如图 13-1 所示。

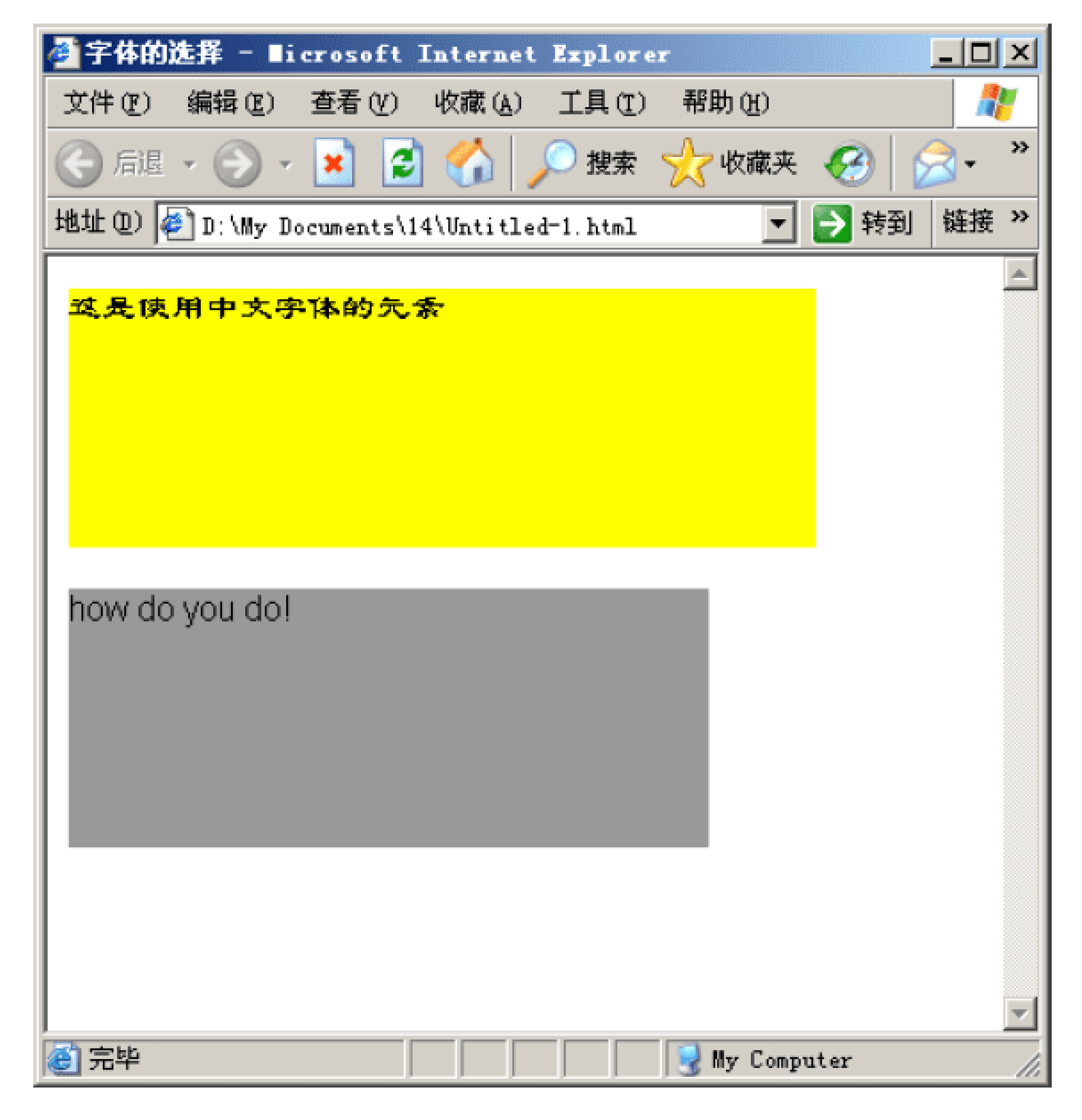


图 13-1　字体属性的显示效果

13.1.2　字体颜色属性 color

字体颜色属性用来定义字体使用的颜色。在定义字体颜色的时候，要注意字体颜色和背景之间的对比，方便读者的阅读。

下面是一个使用字体颜色属性的实例，其代码如下所示。

例程 13-2　example.html

```
<html >
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>css 属性实例</title>
<style>
div
 {
  color:#666666;
  width:350px;
```

```
           height:120px;
           background:#999999;
      }
07   p
      {
         color:red;
         width:300px;
         height:180px;
      }
08   </style>
09   </head>
10   <body>
     <div>注意字体颜色不要和背景颜色太接近</div>
     <p>使用默认背景</p>
11   </body>
12   </html>
```

以上的代码中，06 行在<div>元素中，定义文本的颜色为深灰色，背景颜色为浅灰色。07 行在<p>元素中，定义文本的颜色为红色，背景颜色默认。因为显示与页面颜色一样的白色，这样 p 元素中定义的样式就无法看到。以上代码的显示效果如图 13-2 所示。

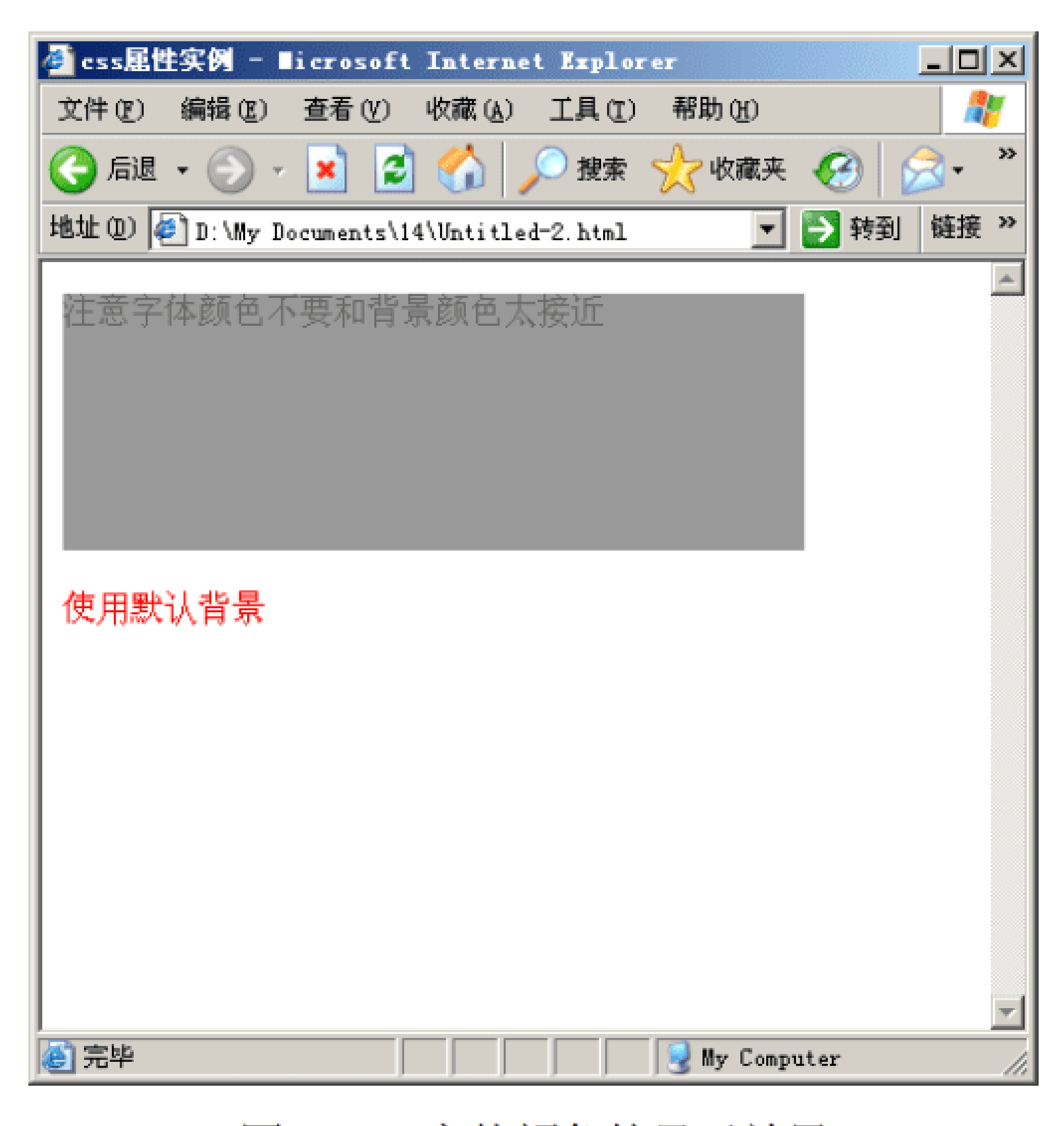


图 13-2　字体颜色的显示效果

从图 13-2 可以看出，字体的颜色和背景颜色之间，要存在一定的对比，才能够更好的阅读。

13.1.3　字体大小属性 font-size

字体大小属性(font-size)用来控制元素中字体的显示尺寸。在字体的大小属性中，可以使用几个属性值，通常使用的是以 px 为单位的长度值，其语法结构如下所示。

```
font-size : xx-small | x-small | small | medium | large | x-large | xx-large | larger | smaller | length;
```

其中各个属性值的含义，如下所示。

- xx-small：定义文本字体为最小。
- x-small：定义文本字体为较小。
- small：定义文本字体为小。
- medium：定义文本字体为正常。
- large：定义文本字体为大。
- x-large：定义文本字体为较大。
- xx-large：定义文本字体为最大。
- larger：相对父元素字体增大。
- smaller：相对父元素字体减小。
- length：把字体设置为一个固定的值。

注意

在以上的属性值中，xx-small、x-small、small、medium、large、x-large、xx-large 等几个值的显示效果，与定义的字体有关。larger、smaller 属性值的显示效果，与父元素中定义的字体大小有关。

下面是一个使用字体大小属性的实例，其代码如下所示。

例程 13-3 example.html

```
<html >
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>字体大小</title>
<style>
.p1
{
font-size:xx-small;}
.p2
{
font-size:x-small;}
.p3
{
font-size:small;}
.p4
{
font-size:medium;}
.p5
{
font-size:large;}
.p6
{
font-size:x-large;}
```

```
.p7
{
font-size:xx-large;}
div{
font-size:36px;}
.p8
{
font-size:smaller;}
.p9
{
font-size:larger;}
</style>
</head>
<body>
<p class="p1">注意字体的大小</p>
<p class="p2">注意字体大小的对比</p>
<p class="p3">注意字体的大小</p>
<p class="p4">注意字体大小的对比</p>
<p class="p5">注意字体的大小</p>
<p class="p6">注意字体大小的对比</p>
<p class="p7">注意字体的大小</p>
<div>注意字体大小的对比
<p class="p8">注意字体的大小</p>
<p class="p9">注意字体大小的对比</p></div>
</body>
</html>
```

以上的代码中，在 9 个<p>元素中，分别定义了不同的字体大小属性值。由于 larger、smaller 属性值，与父元素中定义的字体大小有关，所以使用<div>元素定义了父元素。以上代码的显示效果如图 13-3 所示。

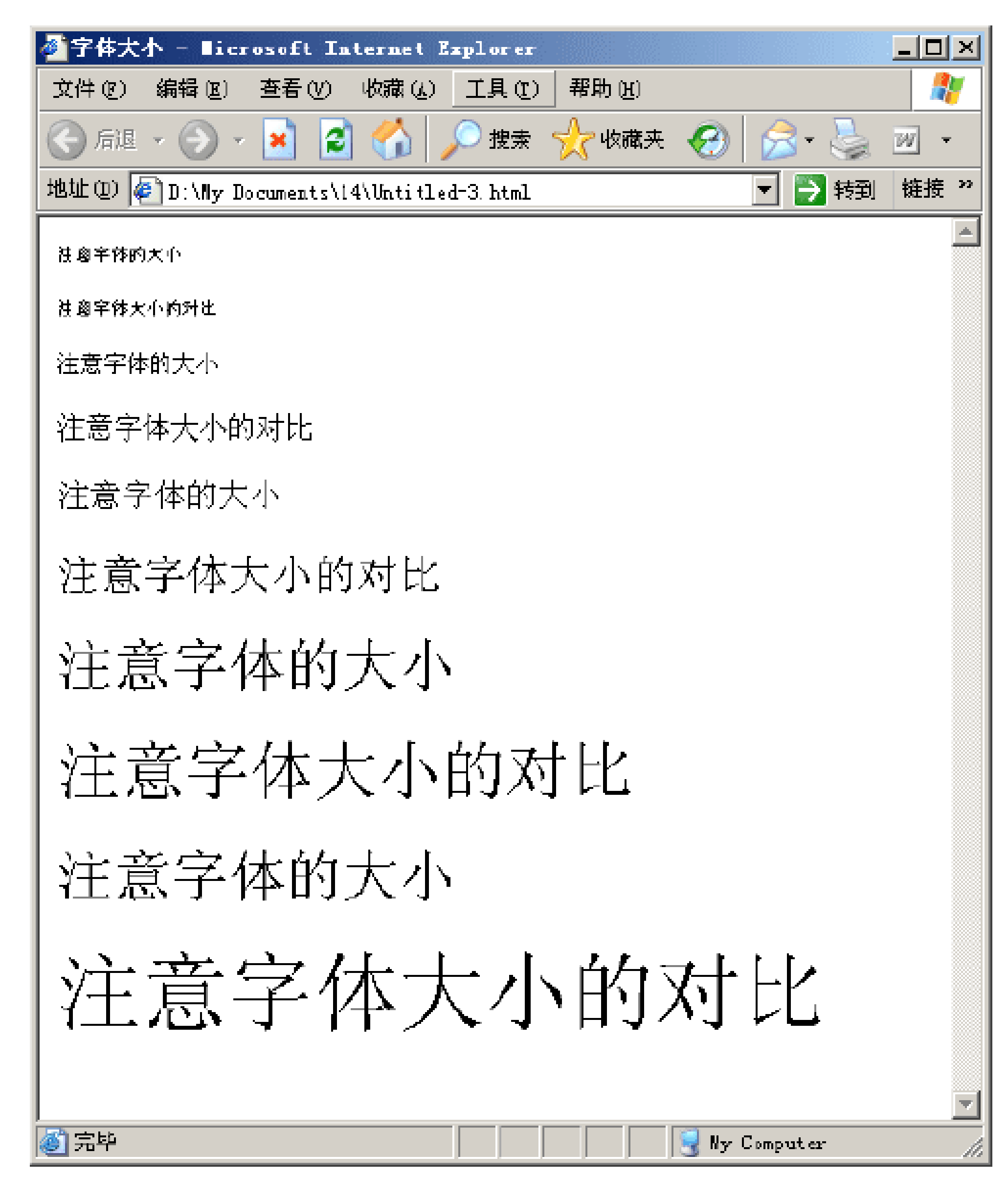


图 13-3　字体大小属性的显示效果

13.1.4　字体样式属性 font-style

字体样式属性(font-style)用来定义字体的显示样式。在字体样式属性中，可以使用几个属性值，包括 norma、italic、oblique。其语法结构如下所示。

```
font-style : normal | italic | oblique;
```

其中各个属性值的含义如下所示。

- normal：定义字体正常显示。
- italic：定义字体使用斜体。
- oblique：定义字体使用倾斜字体。

注意

斜体属性值(italic)用来定义使用斜体字。当字体没有相关的斜体字时，则无法显示斜体效果。倾斜字体(oblique)用来定义字体的倾斜，所有字体都可以使用这个属性定义倾斜效果。斜体字与倾斜的字体，在显示效果上存在着一定的差别。

下面是一个使用字体样式属性的实例，其代码如下所示。

例程 13-4　example.html

```
01  <html >
02  <head>
03  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
04  <title>css 属性实例</title>
05  <style>
06  .p1
  {
    font-style:italic;
    font-size:48px;}
07  .p2
  {
    font-style:normal;
    font-size:48px;}
08  .p3
    {
    font-style:oblique;
    font-size:48px;}
09  </style>
10  </head>
11  <body>
    <p class="p1">这是文本内容</p>
    <p class="p2">这是文本内容</p>
    <p class="p3">这是文本内容</p>
12  </body>
13  </html>
```

以上的代码中，06 行～08 行分别定义了 3 个<p>元素。在每个元素中，同时定义了字体的大小都为 48px，定义了不同的字体样式分别为斜体、正常和倾斜，以便显示效果更加明显。以

上代码的显示效果，如图 13-4 所示。

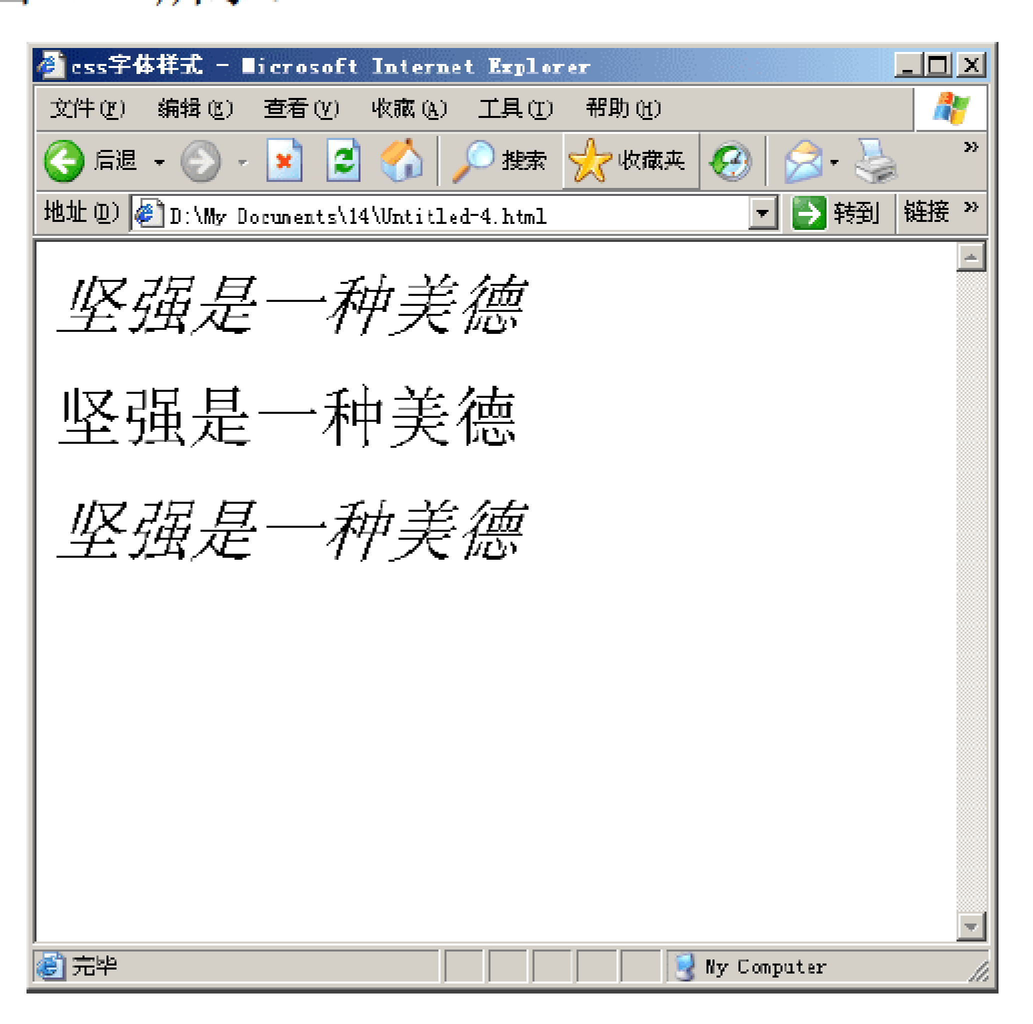


图 13-4　字体样式属性的显示效果

13.1.5　字体加粗属性 font-weight

字体加粗属性(font-weight)用来定义字体是否显示加粗和变细的效果。在字体样式属性中，可以使用两类属性值，一类是用名称命名的属性值，如 blod 等。另一类是用数字命名的属性值，如 200、300 等。其语法结构如下所示。

```
font-weight : normal | bold | bolder | lighter | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900;
```

其中各个属性值的含义如下所示。

- normal：定义字体正常显示。
- bold：定义字体为粗体。
- bloder：以 normal 中定义的字体大小为标准，定义字体比 normal 中字体略大。
- lighter：以 normal 中定义的字体大小为标准，定义字体比 normal 中字体略小。
- 100-900：由 100 至 900，分 9 个层次定义字体的大小。

注意

在使用 100-900 的数字值定义字体加粗属性的时候，可能某些值中定义的加粗效果相同。

下面是一个使用字体加粗属性的实例，其代码如下所示。

例程 13-5　example.html

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>css 字体</title>
<style>
```

```
p
  {
  font-size:18px;}
.p1
  {
  font-weight:normal;}
.p2
  {
  font-weight:bold;}
.p3
  {
  font-weight:bolder;}
.p4
  {
  font-weight:lighter;}
.p5
  {
  font-weight:100;}
.p6
  {
  font-weight:200;}
.p7
  {
    font-weight:800;}
</style>
</head>
<body>
  <p class="p1">谦受益，满招损</p>
  <p class="p2">谦受益，满招损</p>
  <p class="p3">谦受益，满招损</p>
  <p class="p4">谦受益，满招损</p>
  <p class="p5">谦受益，满招损</p>
  <p class="p6">谦受益，满招损</p>
  <p class="p7">谦受益，满招损</p>
</body>
</html>
```

以上的代码中，分别在每个<p>元素中，定义了不同的加粗属性值。同时定义元素中字体的大小为 18px，以便字体的加粗效果能够显示得更加明显。注意使用 bold、bloder 和 800 时候的显示效果。以上代码的显示效果，如图 13-5 所示。

可以看到当使用数字加粗时，数字值要足够大才有效果。

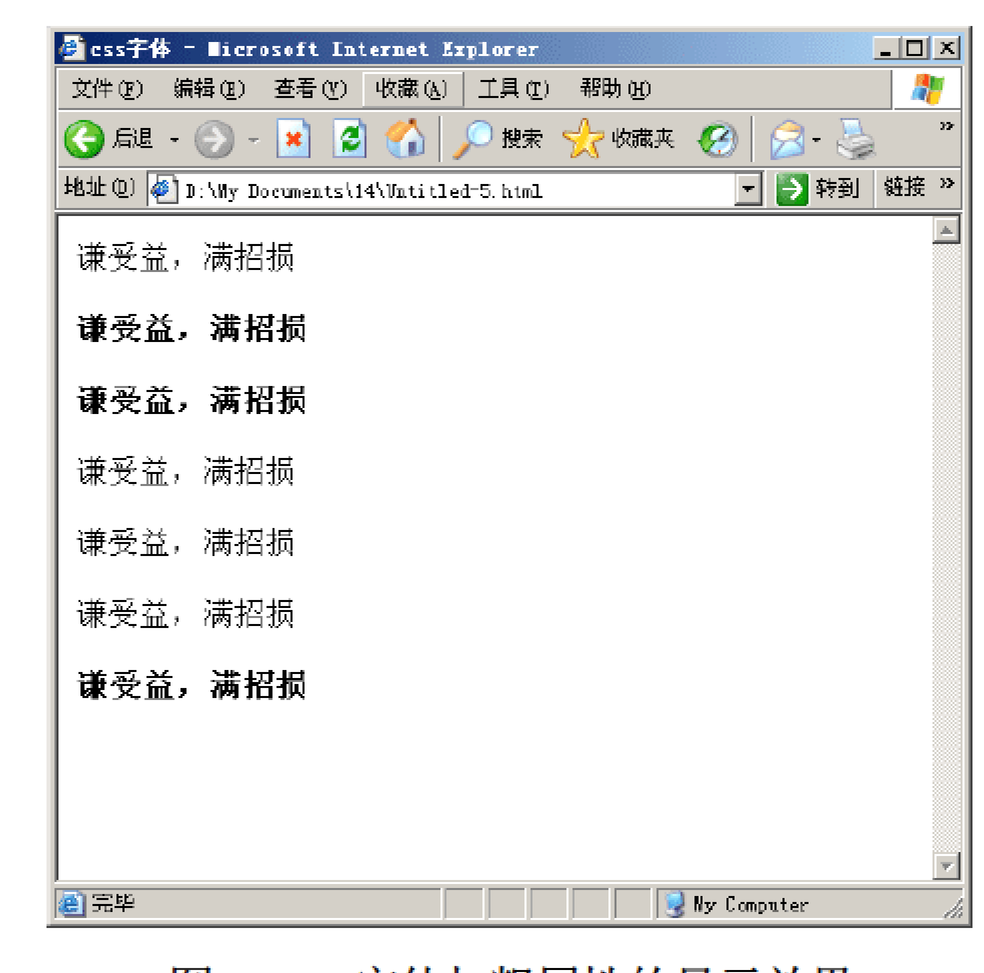


图 13-5　字体加粗属性的显示效果

13.1.6　字体修饰属性 text-decoration

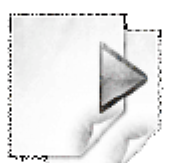
字体修饰属性(text-decoration)用来定义字体的修饰效果，如下划线等。在字体修饰属性中，可以

使用 5 个属性值，分别是 none、underline、blink、overline 和 line-through。其语法结构如下所示。

```
text-decoration : none | underline | blink | overline | line-through;
```

其中各个属性值的含义如下所示。

- none：字体无修饰效果。
- overline：定义字体修饰效果为上划线。
- line-through：定义字体修饰效果为删除线。
- underline：定义字体的修饰效果为下划线。
- blink：定义字体修饰效果为闪烁(IE 6.0 中不支持此效果)。



注意

可以同时给某个元素定义多个修饰属性。

下面是一个使用字体样式属性的实例，其代码如下所示。

例程 13-6 example.html

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>css 属性实例</title>
<style>
.p1
  {
  text-decoration:none;
  }
.p2
  {
  text-decoration:overline;
  }
.p3
  {
  text-decoration:line-through;
  }
.p4
  {
  text-decoration:underline;
  }
</style>
</head>
<body>
<p class="p1">谦受益满招损</p>
<p class="p2">谦受益满招损</p>
<p class="p3">谦受益满招损</p>
<p class="p4">谦受益满招损</p>
</body>
</html>
```

以上的代码中，定义了 4 个<p>元素。在每个元素中，分别定义了无修饰、上划线、删除线和下划线。以上代码的显示效果，如图 13-6 所示。

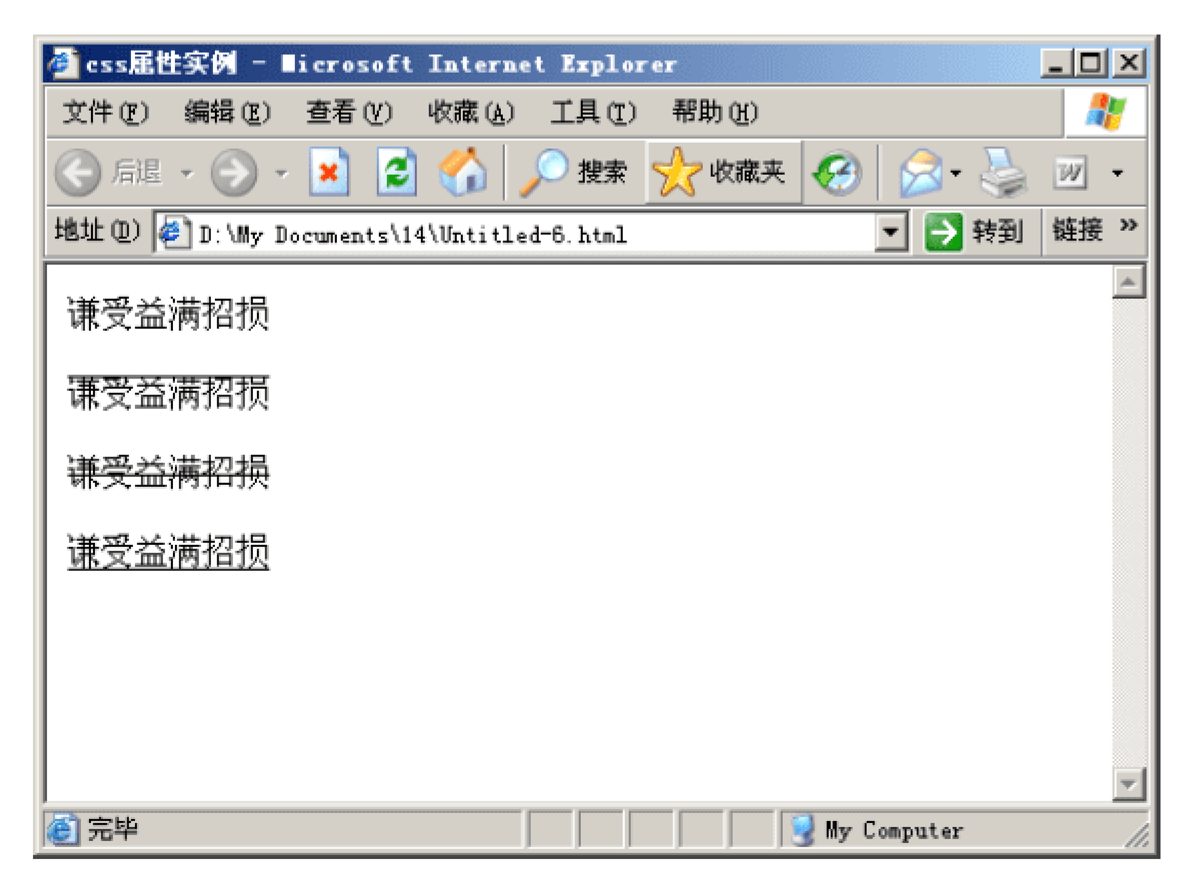


图 13-6　字体修饰属性的显示效果

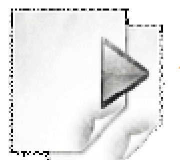
13.1.7　字体下划线位置属性 text-underline-position

字体下划线位置属性(text-underline-position)用来定义下划线的位置。在字体下划线位置属性中，可以使用两个属性值，分别是 below 和 above。其语法结构如下所示。

```
text-underline-position : below | above;
```

其中各个属性值的含义如下所示。

- above：下划线在文本的上面。
- below：下划线在文本的下面。



注意

在使用字体下划线位置属性的时候，一定要先定义下划线属性。

下面是一个使用字体下划线位置属性的实例，其代码如下所示。

例程 13-7　example.html

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>下划线的位置</title>
<style>
.p1
{
text-decoration:underline;
text-underline-position:above;
}
.p2
```

```
{
text-decoration:underline;
text-underline-position:below;
}
p
{
 font-size:50px;
}
</style>
</head>
<body>
<p class="p1">吃一堑长一智</p>
<p class="p2">吃一堑长一智</p>
</body>
</html>
```

以上的代码中，定义了 2 个<p>元素。在每个元素中，分别定义了下划线在上和下两个位置。以上代码的显示效果，如图 13-7 所示。



图 13-7　字体修饰属性的显示效果

13.1.8　小型大写字母属性 font-variant

小型大写字母属性(font-variant)用来定义英文中是否显示小型大写字母的效果。小型大写字母是介于大写字母和小写字母之间的一种字母显示效果。在小型大写字母属性中，可以使用两个属性值，分别是 normal 和 above。其语法结构如下所示。

```
font-variant : normal | small-caps;
```

其中各个属性值的含义如下所示。

- normal：文本显示正常效果。
- small-caps：文本以小型大写字母效果显示。

注意

在使用小型大写字母属性的时候，文本中要有小写字母才能够显示出效果，否则将显示大写字母的效果。因为中文中没有小型大写字母的概念，所以对中文不起作用。

下面是一个使用小型大写字母属性的实例，其代码如下所示。

例程 13-8 example.html

```
01 <html xmlns="http://www.w3.org/1999/xhtml">
02 <head>
03 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
04 <title>css 属性实例</title>
05 <style>
06 div
   {
    font-variant:small-caps;
   font-size:48px;
   }
07 p
   {
    font-size:48px;
   }
08 </style>
09 </head>
10 <body>
   <div>How are you!</div>
   <p>How are you!</p>
11 </body>
12 </html>
```

以上的代码中，06 行在<div>元素中定义文本的显示效果为小型大写字母。07 行在<p>元素中，定义文本为普通文本。同时定义了文本的大小为 48px，目的是使字母的显示效果更加明显。以上代码的显示效果如图 13-8 所示。

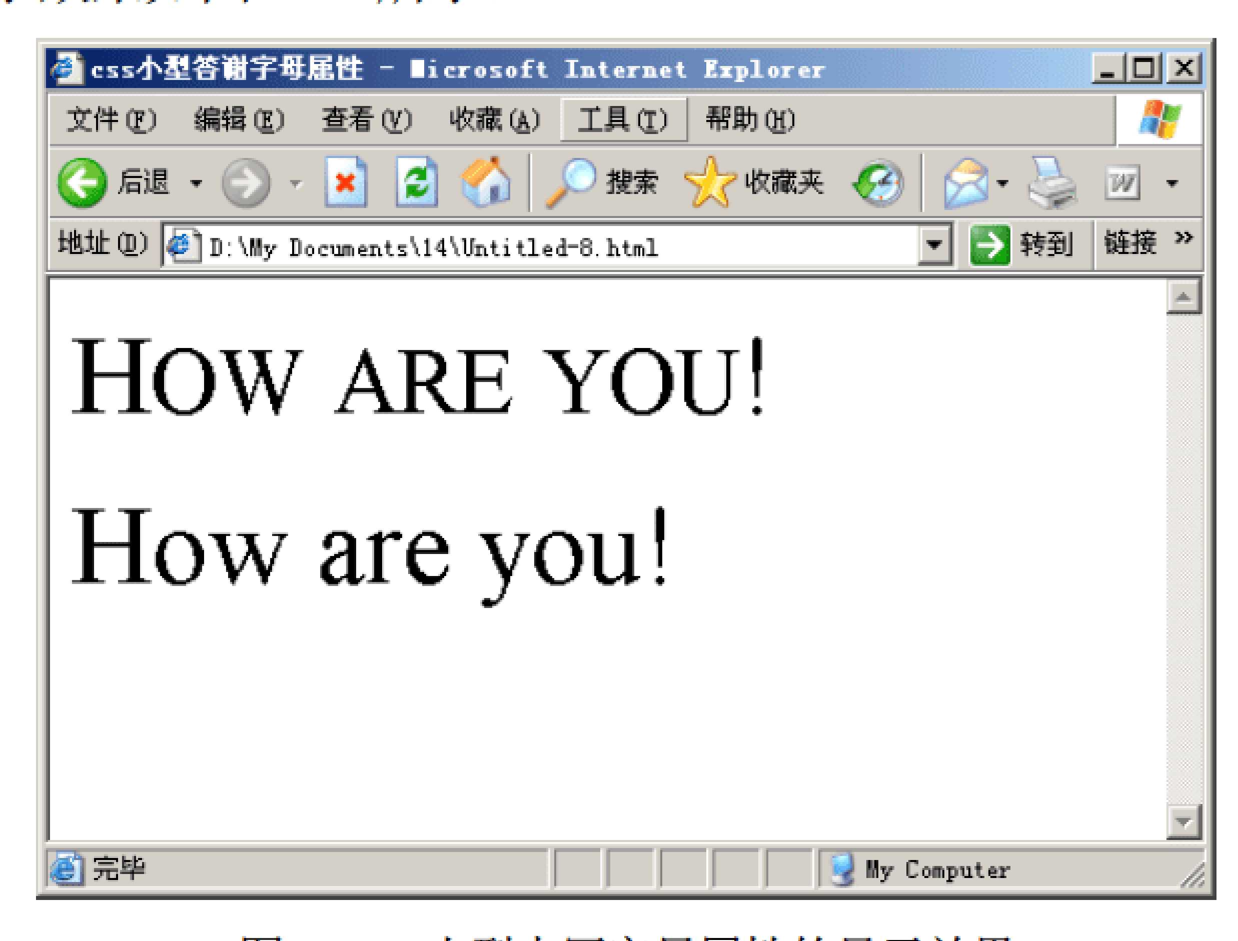


图 13-8 小型大写字母属性的显示效果

13.1.9　转换大小写属性 text-transform

转换大小写属性(text-transform)用来定义英文字母大小写之间转换的效果。在转换大小写属性中，可以使用 4 个属性值，分别是 none、capitalize、uppercase 和 lowercase。其语法结构如下所示。

```
text-transform : none | capitalize | uppercase | lowercase;
```

其中各个属性值的含义如下所示。

- none：字母显示效果不发生转换。
- capitalize：文本中首字母大写。
- uppercase：字母转换为大写字母。
- lowercase：字母转换为小写字母。



注意

由于在中文中无大小写的问题，所以转换大小写属性对中文文本不起作用。

下面是一个使用转换大小写属性的实例，其代码如下所示。

例程 13-9　example.html

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>转换大小写属性</title>
<style>
.p1
{
text-transform:capitalize;
}
.p2
{
text-transform:lowercase;
}
.p3
{
text-transform:uppercase;
}
.p4
{
text-transform:none;
}
p
{
font-size:48px;
}
```

```
</style>
</head>
<body>
<p class="p1">How old are you!</p>
<p class="p2">How old are you!</p>
<p class="p3">How old are you!</p>
<p class="p4">How old are you!</p>
</body>
</html>
```

以上的代码中，定义了 4 个<p>元素。在每个元素中，定义了不同的大小写转换值。第 1 个<p>元素中，定义了首字母大写。第 2 个<p>元素中，定义了所有字母小写。第 3 个<p>元素中，定义了所有字母大写。第 4 个<p>元素中，定义了字母正常显示。以上代码的显示效果如图 13-9 所示。

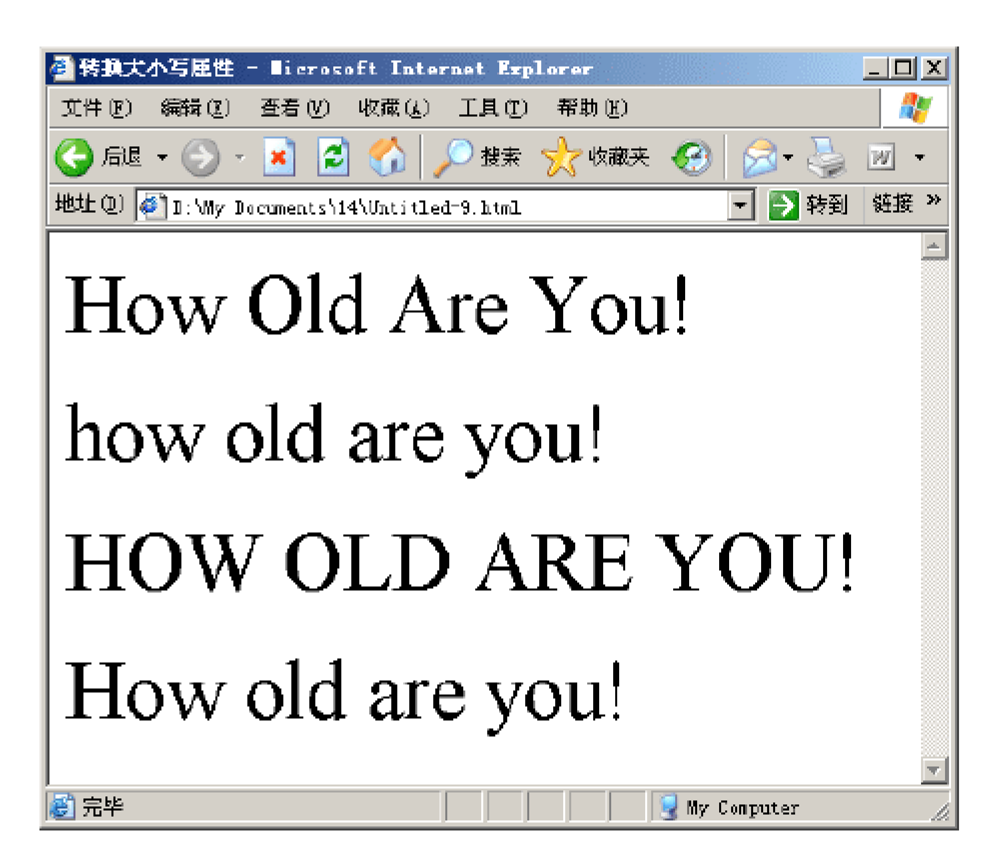


图 13-9　转换大小写属性的显示效果

13.1.10　字母间隔属性 letter-spacing

字母间隔属性(letter-spacing)用来定义字母(或中文文字)之间的间隔大小。在字母间隔属性中，可以使用两个属性值，分别是 normal 和 length。其语法结构如下所示。

```
letter-spacing : normal | length;
```



注意

在字母间隔属性中，不能使用百分比值。

下面是一个使用字母间隔属性的实例，其代码如下所示。

例程 13-10　example.html

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title> 字母间隔属性</title>
<style>
.spacing1
{
letter-spacing:4em;
margin:20px;}
.spacing2
{
letter-spacing:20px;}
```

```
div,p
{
font-size:40px;}
</style>
</head>
<body>
<div class="spacing1">庆祝中华人民共和国 60 周年</div>
<div>庆祝中华人民共和国 60 周年</div>
<p class="spacing2">How are you!</p>
<p>How are you!</p>
</body>
</html>
```

以上的代码中的第 1 个<spacing>元素中，定义元素中文本的间隔属性值为 4em，同时定义了边界属性，用来分隔下面相邻的元素。第 2 个<spacing>元素中，只定义了文本字体的大小，用来和第一个<spacing>元素中的内容作对照。<p>和<div>元素中定义文本的间隔属性值为 40px，用来做对照。以上代码的显示效果如图 13-10 所示。

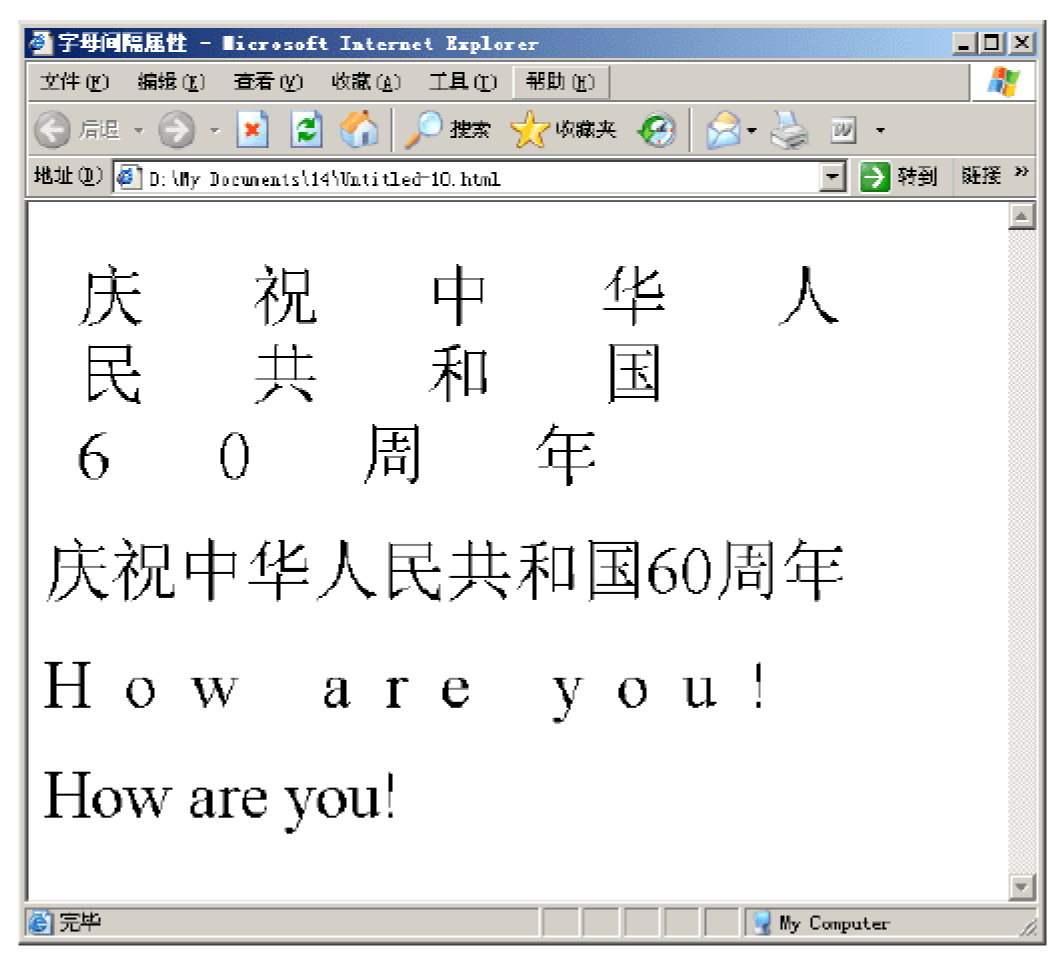


图 13-10　文本的间隔属性的显示效果

13.1.11　单词间隔属性 word-spacing

单词间隔属性(word-spacing)用来定义英文单词之间的间隔。在单词间隔属性中，可以使用两个属性值，分别是 normal 和 length。其语法结构如下所示。

```
word-spacing : normal | length;
```



注意

由于在中文中没有单词这个语言单位，所以单词间隔属性对中文文本不起作用。

下面是一个使用单词间隔属性的实例，其代码如下所示。

例程 13-11 example.html

```
01	<html xmlns="http://www.w3.org/1999/xhtml">
02	<head>
03	<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
04	<title>单词间隔属性</title>
05	<style>
06	.spacing1
	{
	word-spacing:5em;
	margin:0 0 20px 0;
	}
07	.spacing2
	{
	word-spacing:30px;
	}
08	div,p
	{
	font-size:45px;
	}
09	</style>
10	</head>
11	<body>
	<div class="spacing1">这个属性对中文是无用的</div>
	<div>这个属性对中文无用</div>
	<p class="spacing2">How are you!</p>
	<p>How are you!</p>
12	</body>
13	</html>
```

以上的代码中，06 和 07 行分别在第 1 个和第 2 个<spacing>元素中，定义了单词间隔属性值。同时 08 行定义了<div>元素和<p>元素作为显示效果的参照。以上代码运行后可以看到中文的显示不受任何影响，而英文各个单词之间间开了一定的距离。运行的显示效果如图 13-11 所示。

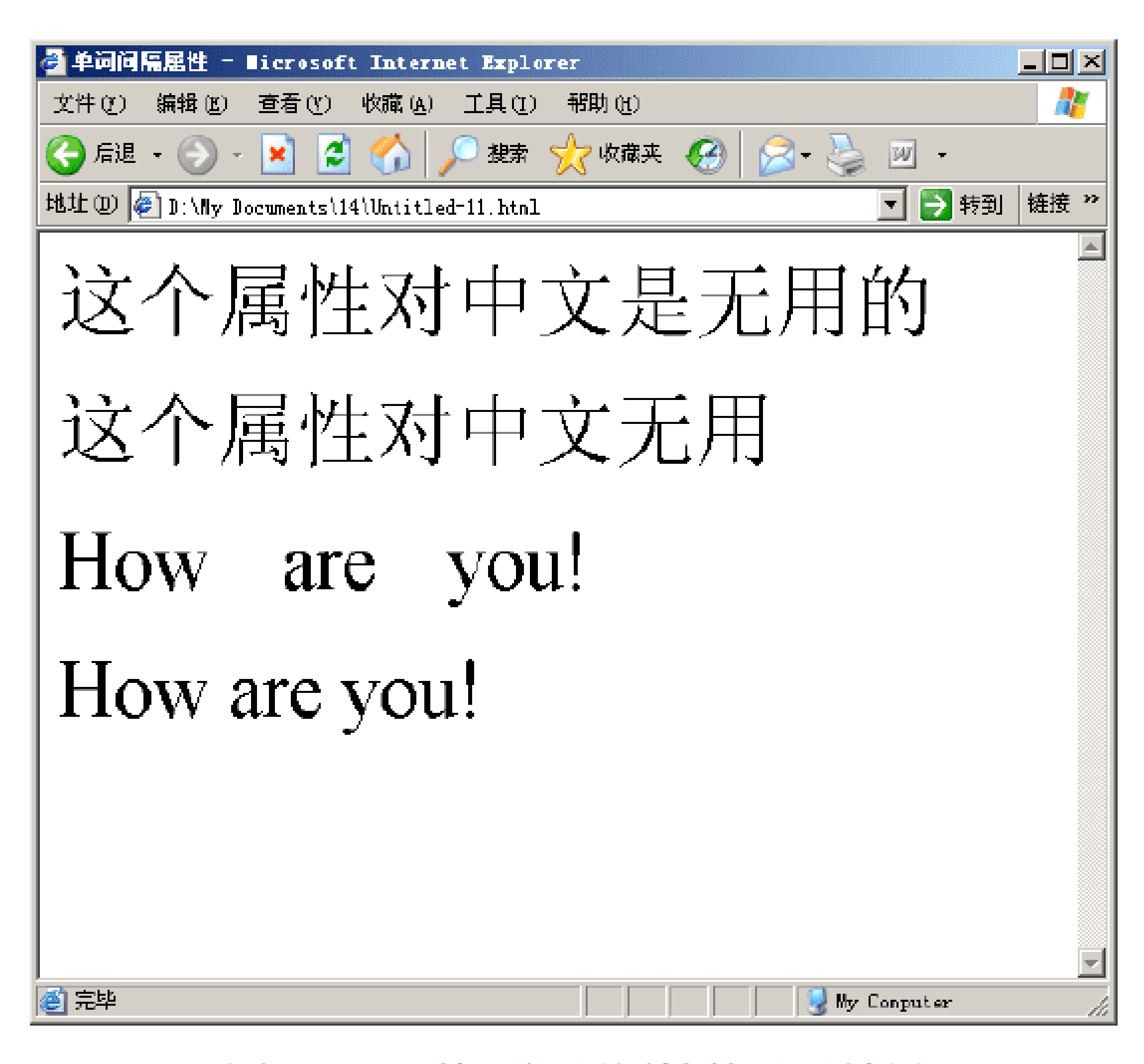


图 13-11　单词间隔属性的显示效果

13.1.12　行高属性 line-height

行高属性(line-height)用来定义文本中行高的大小。在行高属性中，可以使用 3 个属

性值，分别是 normal、length 和 percent。其语法结构如下所示。

```
Line-height : normal | length | percent;
```



注意

在行高属性中，使用百分比值，其最终取值以文本中字体的高度为基准。

下面是一个使用行高属性的实例，其代码如下所示。

例程 13-12 example.html

```
01   <html xmlns="http://www.w3.org/1999/xhtml">
02   <head>
03   <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
04   <title>css 行高属性</title>
05   <style>
06   div
{
     line-height:normal;
     font-size:2em;
     width:500px;
     height:100px;
}
07   p
{
     line-height:200%;
     font-size:36px;
     width:500px;
     height:100px;
}
08   </style>
09   </head>
10   <body>
     <div>这是一个关于行高属性的实例，注意换行后文本的显示效果。</div>
     <p>这是一个关于行高属性的实例，注意换行后文本的显示效果。</p>
11   </body>
12   </html>
```

以上的代码中，06 行在<div>元素中定义行高为正常。07 行在<p>元素中定义行高为 200%。同时定义了元素的大小，用来产生换行效果。定义了元素的字体大小，用来使效果显示得更加明确。以上代码的显示效果如图 13-12 所示。

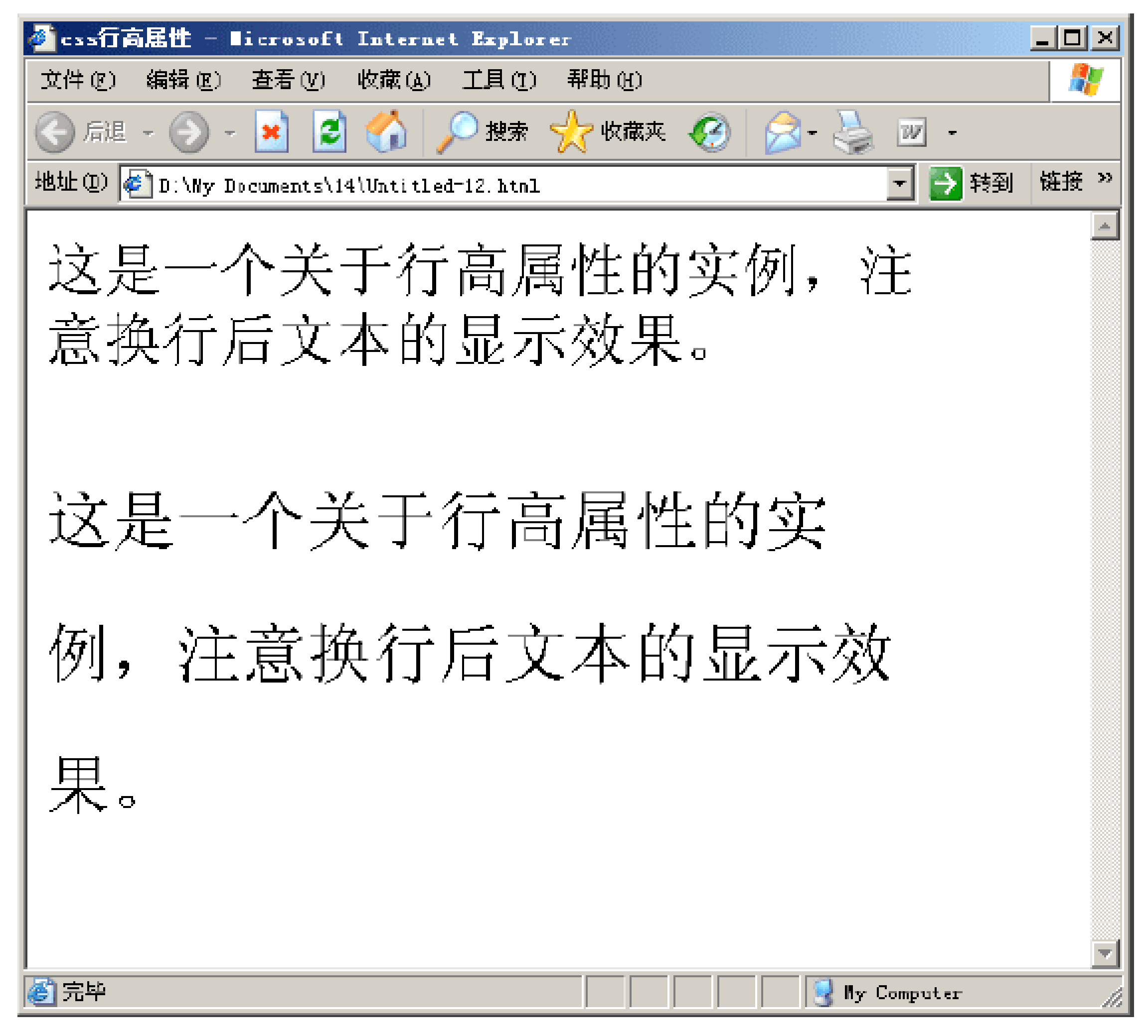


图 13-12　行高属性的显示效果

可以看到<p>元素中的两行之间有了距离。

13.1.13　字体综合属性 font

字体综合属性(font)用来统一定义字体的各种属性值。在字体综合属性中，可以使用以上几个小节中讲解的部分属性。其语法结构如下所示。

```
font : font-family   font-style   font-variant   font-weight   font-size line-height;
```

在字体的综合属性中，每个属性值之间使用空格分隔。如果同时使用了字体大小属性和行高属性，要使用“字体大小/行高”的格式。

注意

字体的综合属性中，每个属性之间是并列的关系，交换部分属性值之间的顺序，并不影响属性的显示效果。在使用中文字体的时候，最好将中文字体内容定义在属性值的最后面，否则可能会带来显示上的问题。

下面是一个使用字体综合属性的实例，其代码如下所示。

例程 13-13　example.html

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title> 字体综合属性</title>
<style>
p
{
font:24px/200% blod italic   "黑体";
```

```
         width:300px;
         height:100px;
       }
07   </style>
08   </head>
09   <body>
     <p>这是一个关于字体综合属性的实例，注意文本的显示效果。</p>
10   </body>
11   </html>
```

以上的代码中，06 行中使用字体综合属性统一定义了元素中字体的大小、使用的字体、行高、加粗、样式等属性，同时定义了元素的大小，以便显示换行的效果。以上代码的显示效果如图 13-13 所示。

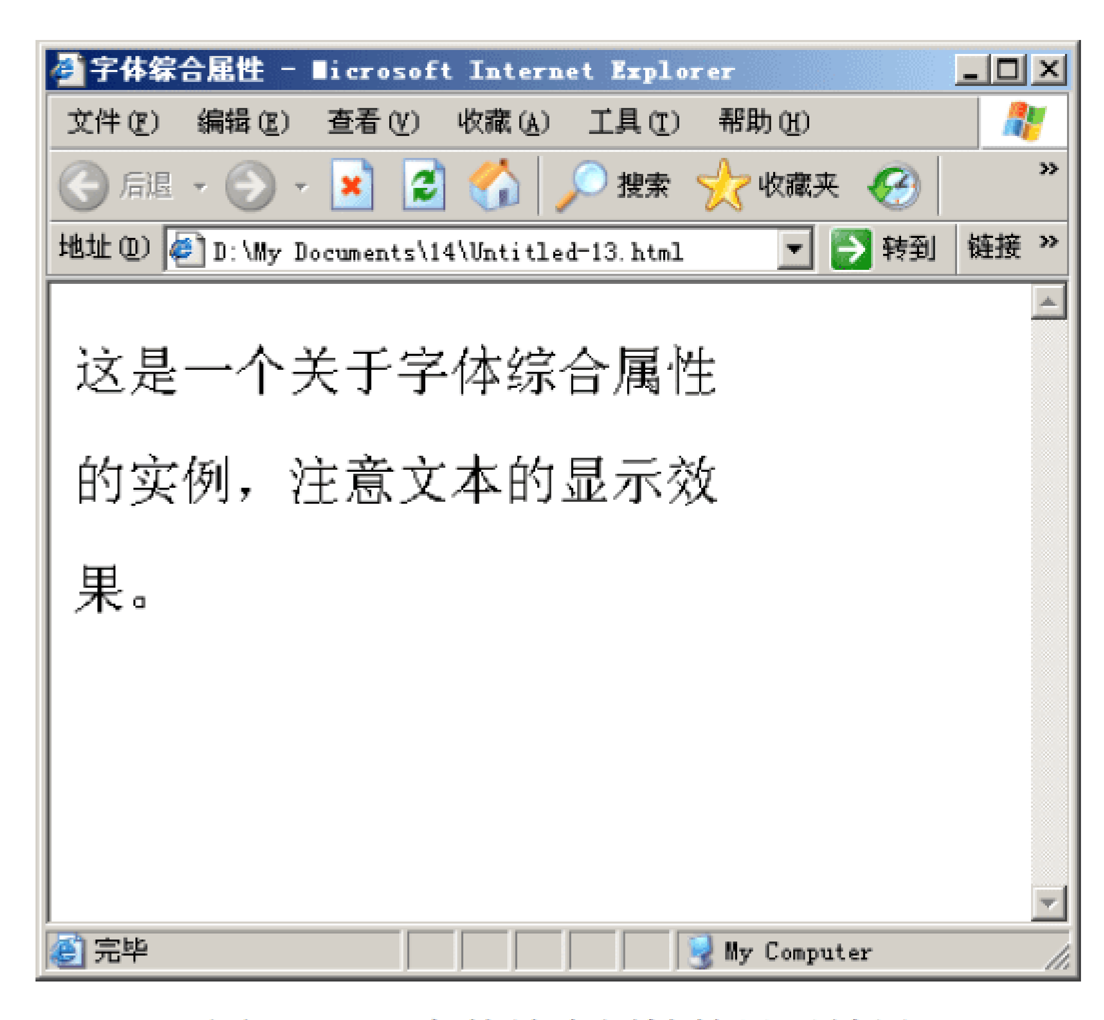


图 13-13　字体综合属性的显示效果

13.2 控制文本的显示

在 CSS 中文本的控制是指控制文本的缩进、对齐、空白、显示方向等内容。在实际制作网页时，文本的控制对整个页面的排版起着决定性的作用。好的文本效果，不但美观，更加便于读者的阅读。下面进行详细讲解。

13.2.1 文本缩进属性 text-indent

文本缩进属性(text-indent)用来定义文本段落中段首的缩进效果。在文本的缩进属性中，使用的属性值为长度值，其语法结构如下所示。

```
text-indent : length;
```



注意

使用文本的缩进属性，只对元素中的段落产生影响。

下面是一个使用文本缩进属性的实例，其代码如下所示。

例程 13-14 example.html

```
01  <html xmlns="http://www.w3.org/1999/xhtml">
02  <head>
03  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
04  <title>文本缩进属性</title>
05  <style>
06  p
  {
    text-indent:3em;
    font-size:24px;
    width:400px;
    height:100px;
  }
07  </style>
08  </head>
09  <body>
    <p>这是一个关于文本缩进属性的实例，注意这段话开始部分的显示效果。</p>
    <p>这是一个关于文本缩进属性的实例，注意这段话开始部分的显示效果。</p>
10  </body>
11  </html>
```

以上的代码中，06 行在<p>元素中定义了文本的缩进属性值为 3em，同时定义文本内容中字体的大小为 24 像素，元素的宽度为 400 像素、高度为 100 像素以便于文本内容的显示。其显示效果如图 13-14 所示。

从图 13-14 可以看出，在父元素中定义了文本的缩进属性，以后各个子元素中将继续继承这个缩进属性值。

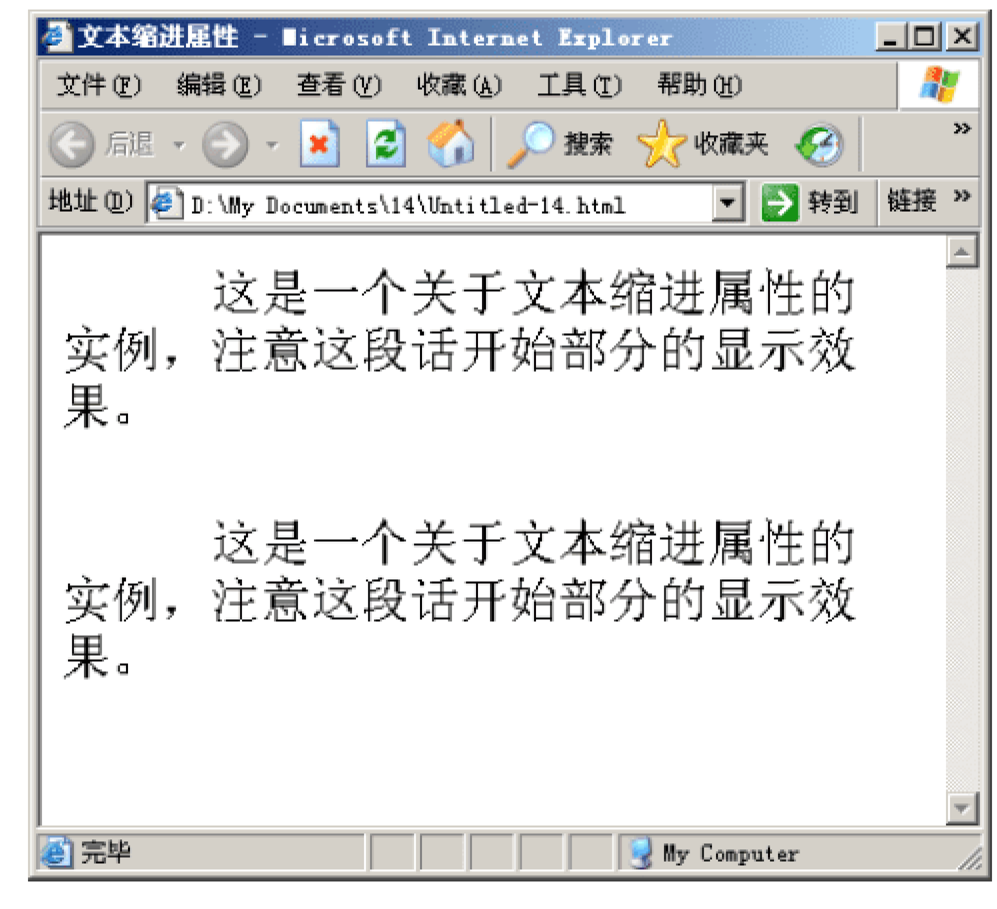


图 13-14 文本缩进属性的显示效果

13.2.2 文本空白属性 white-space

文本空白属性(white-space)用来把文本中使用空格、换行等空白元素的内容显示出来。在文本的空白属性中，可以使用 3 个属性值，分别是 normal、pre 和 nowrap，其语法结构如下所示。

```
white-space : normal | pre | nowrap;
```

其中各个属性值的含义，如下所示。

- normal：定义空白内容，按照默认的方式显示。
- pre：定义空白内容，按照原有的方式显示。
- nowrap：定义内容同行显示。

下面是一个使用文本空白属性的实例，其代码如下所示。

例程 13-15 example.html

```
01  <html xmlns="http://www.w3.org/1999/xhtml">
02  <head>
03  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
04  <title>文本空白属性</title>
05  <style>
06  .p1
    {
    white-space:nowrap;
    }
07  .p2
    {
   white-space:normal;
    }
08  .p3
    {
    white-space:pre;
    }
09  p
    {
    width:300px;
    border:2px solid #000000;
    padding:10px;
    }
10  </style>
11  </head>
12  <body>
    <p class="p1">这段文本内中，使用了空格      和换行
    等内容，要注意他们在页面中的显示方式</p>
    <p class="p2">这段文本内中，使用了空格      和换行
    等内容，要注意他们在页面中的显示方式</p>
    <p class="p3">这段文本内中，使用了空格      和换行等内
    容，要注意他们在页面中的显示方式</p>
13  </body>
14  </html>
```

以上的代码中，06 到 08 行在 3 个<p>元素中分别定义了不同的文本空白属性值，同时 09 行定义了<p>元素的宽度、边框、补白等属性，目的是使显示效果更加明显。代码运行后的显示效果如图 13-15 所示。

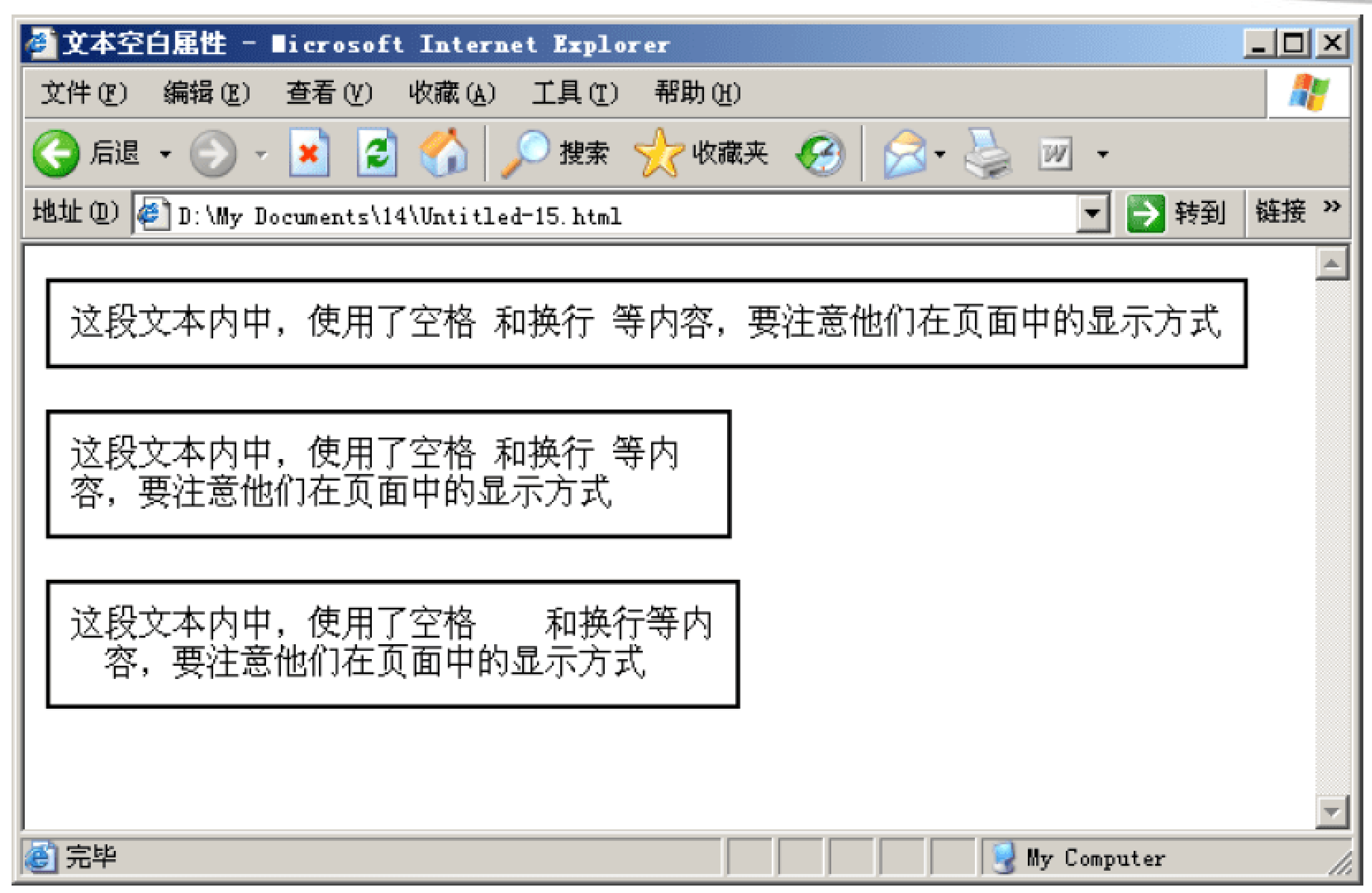


图 13-15　文本空白属性的显示效果

从图 13-15 可以看出，使用 pre 和 nowrap 值的元素，当内容宽度超出元素宽度时，都会强行改变元素的显示宽度进行换行。

13.2.3　文本溢出属性 text-overflow

当文本内容超出元素大小的时候，就要用到文本溢出属性(text-overflow)来定义这些文本的显示效果。在文本溢出属性中，可以使用两个属性值，分别是 clip 和 ellipsis。其语法结构如下所示。

```
text-overflow : clip | ellipsis;
```

其中每个属性值的含义，如下所示。

- clip：裁减但不显示省略标记。
- ellipsis：当文本溢出时，显示省略标记。

注意

只有当元素中使用了 overflow 属性，并且文本空白属性值为 nowrap 的时候，文本的溢出属性才能够显示效果。

下面是一个使用文本溢出属性的实例，其代码如下所示。

例程 13-16　example.html

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>文本溢出属性</title>
<style>
```

```
06  div
    {
        text-overflow:ellipsis;
        overflow:hidden;
        white-space:nowrap;
        font-size:36px;
        width:400px;
    }
07  p
    {
        text-overflow:clip;
        overflow:hidden;
        white-space:nowrap;
        font-size:36px;
        width:400px;
    }
08  </style>
09  </head>
10  <body>
    <div>这是一个关于文本溢出属性的实例，注意文本中最后部分的显示效果。</div>
    <p>这是一个关于文本溢出属性的实例，注意文本中最后部分的显示效果</p>
11  </body>
12  </html>
```

以上的代码中，06 行在<div>元素中定义了文本溢出属性值为显示省略标记，同时定义了溢出内容显示效果为隐藏，文本显示效果为同行显示。07 行在<p>元素中定义文本溢出属性值为裁剪。代码运行后的效果如图 13-16 所示。

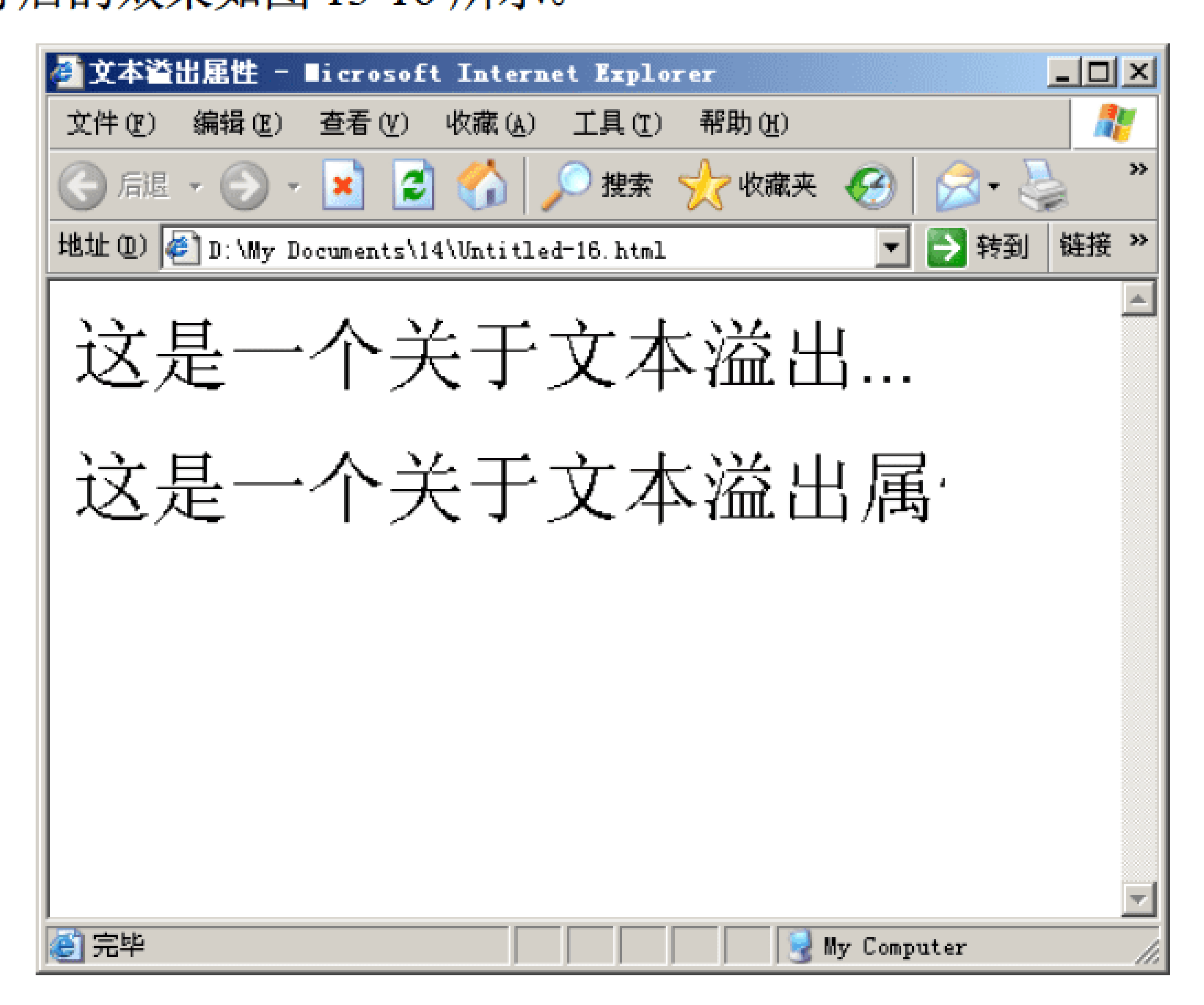


图 13-16　文本溢出属性的显示效果

从图 13-16 可以看出，在溢出属性中使用隐藏值(hidden)的时候，就会显示文本溢出属性中

定义的省略效果。如果使用的是可见值(visible)，则这些效果将不会被显示出来。

13.2.4 水平对齐属性 text-align

水平对齐属性(text-align)用来定义元素的水平对齐效果。在水平对齐属性中，可以使用 4 个属性值，分别定义元素内容的左对齐、居中对齐、右对齐和两边对齐，其语法结构如下所示。

```
text-align : left | right | center | justify;
```

其中每个属性的含义如下所示。

- left：定义元素内容左侧对齐。
- right：定义元素内容右侧对齐。
- center：定义元素内容居中对齐。
- justify：定义元素内容两边对齐(目前浏览器还不支持该属性)。

下面是一个使用水平对齐属性的实例，其代码如下所示。

例程 13-17 example.html

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title> 水平对齐属性</title>
<style>
p
  {
    font-size:24px;
    width:400px;
    height:60px;
    border:4px solid red;
  }
.p1
  {
    text-align:left;
  }
.p2
  {
    text-align:center;
  }
.p3
  {
   text-align:right;
  }
.p4
  {
    text-align:justify;
  }
</style>
</head>
```

```
13  <body>
    <p class="p1">注意文本的对齐方式</p>
    <p class="p2">注意文本的对齐方式</p>
    <p class="p3">注意文本的对齐方式</p>
    <p class="p4">注意文本的对齐方式</p>
14  </body>
15  </html>
```

以上的代码中，07 行～10 行中在 4 个<p>元素中分别定义了不同的水平对齐属性值。同时 06 行中定义了元素的宽度、高度、边框等属性，目的是更好地显示对齐的效果。代码运行后，其显示效果如图 13-17 所示。

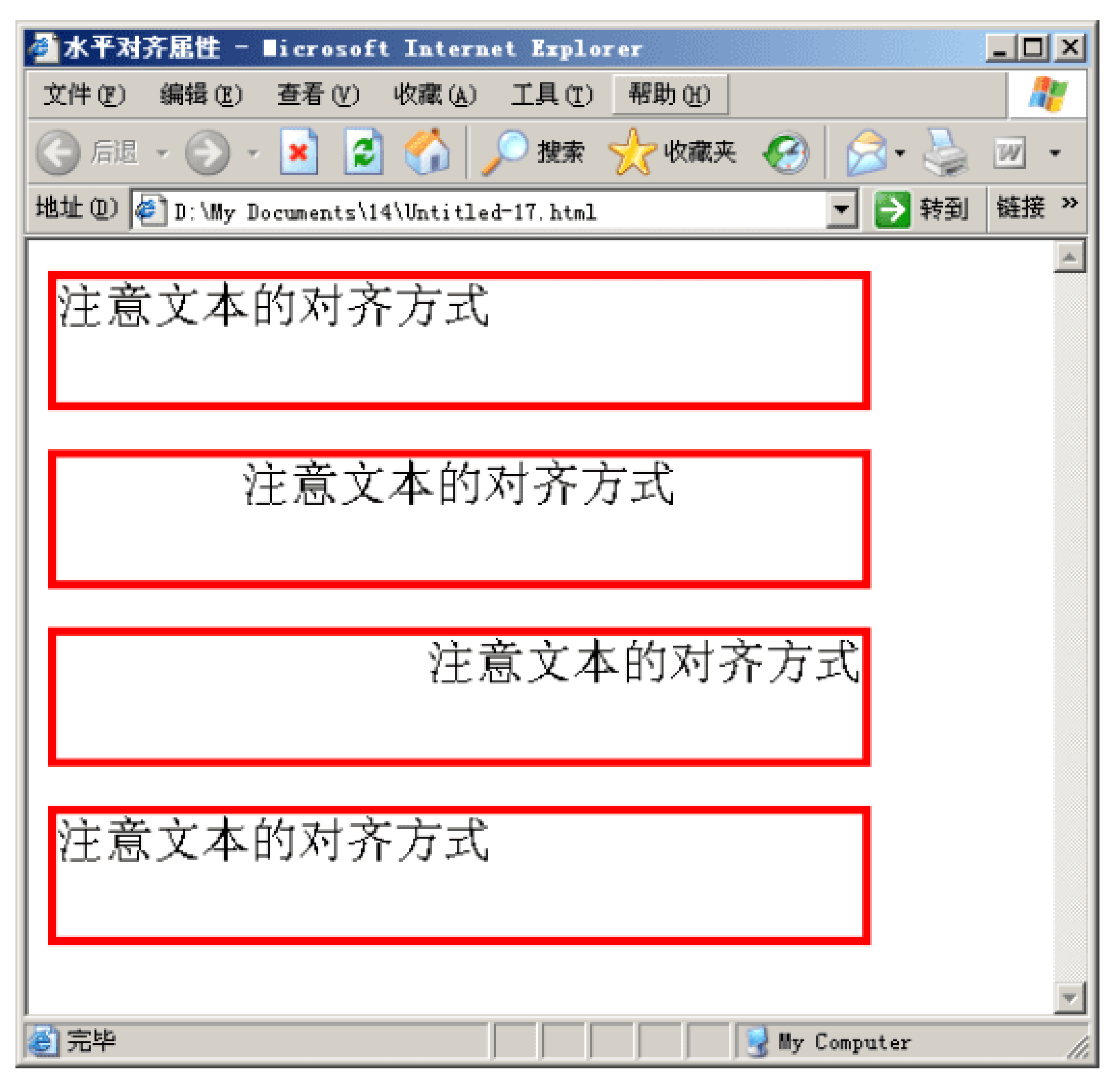


图 13-17　定义水平对齐属性后的显示效果

13.2.5　垂直对齐属性 vertical-align

垂直对齐属性(vertical-align)用来定义元素的垂直对齐效果。在垂直对齐属性中，可以使用 10 个属性值，其语法结构如下所示。

```
vertical-align : auto | baseline | sub | super | top | text-top | middle | bottom | text-bottom | length;
```

其中每个属性的含义如下所示。

- auto：自动对齐元素(和 layout-flow 属性值有关)。
- baseline：定义内容与基线对齐。
- sub：定义元素内容与上标对齐。
- super：定义元素内容与下标对齐。
- top：定义元素内容与顶部对齐。
- text-top：定义元素内容与文本顶部对齐。

- middle：定义元素内容居中对齐。
- bottom：定义元素内容与底部对齐。
- text-bottom：定义元素内容与文本底部对齐。
- length：使用长度值(浏览器未支持该属性)。



注意

垂直对齐属性只能应用于内联元素，在块元素中使用垂直对齐属性将无法发挥作用。在使用垂直布局属性布局的时候，要十分注意这一点。

下面是一个使用垂直对齐属性的实例，其代码如下所示。

例程 13-18 example.html

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>css 属性实例</title>
<style>
.img1
{
vertical-align:baseline;
}
.img2
{
vertical-align:bottom;
}
.img3
{
vertical-align:middle;
}
.img4
{
vertical-align:sub;
}
.img5
{
vertical-align:super;
}
.img6
{
vertical-align:text-bottom;
}
.img7
{
 vertical-align:text-top;
}
.img8
{
```

```
        vertical-align:top;
        }
14      p
        {
        font-size:18px;
        width:400px;
        height:45px;
        border:2px solid #000000;
        }
15      </style>
16      </head>
17      <body>
        <p>这是图像内容<img class="img1" src="images/greyscale_006.gif" alt="pic" />这是文本内容。</p>
        <p>这是图像内容<img class="img2" src="images/greyscale_006.gif" alt="pic" />这是文本内容。</p>
        <p>这是图像内容<img class="img3" src="images/greyscale_006.gif" alt="pic" />这是文本内容。</p>
        <p>这是图像内容<img class="img4" src="images/greyscale_006.gif" alt="pic" />这是文本内容。</p>
        <p>这是图像内容<img class="img5" src="images/greyscale_006.gif" alt="pic" />这是文本内容。</p>
        <p>这是图像内容<img class="img6" src="images/greyscale_006.gif" alt="pic" />这是文本内容。</p>
        <p>这是图像内容<img class="img7" src="images/greyscale_006.gif" alt="pic" />这是文本内容。</p>
        <p>这是图像内容<img class="img8" src="images/greyscale_006.gif" alt="pic" />这是文本内容。</p>
18      </body>
19      </html>
```

以上的代码中，在 8 个<img>元素中分别定义了不同的垂直对齐属性值。同时定义了元素的宽度、高度、边框等属性，目的是更好地显示对齐的效果。代码运行后，其显示效果如图 13-18 所示。

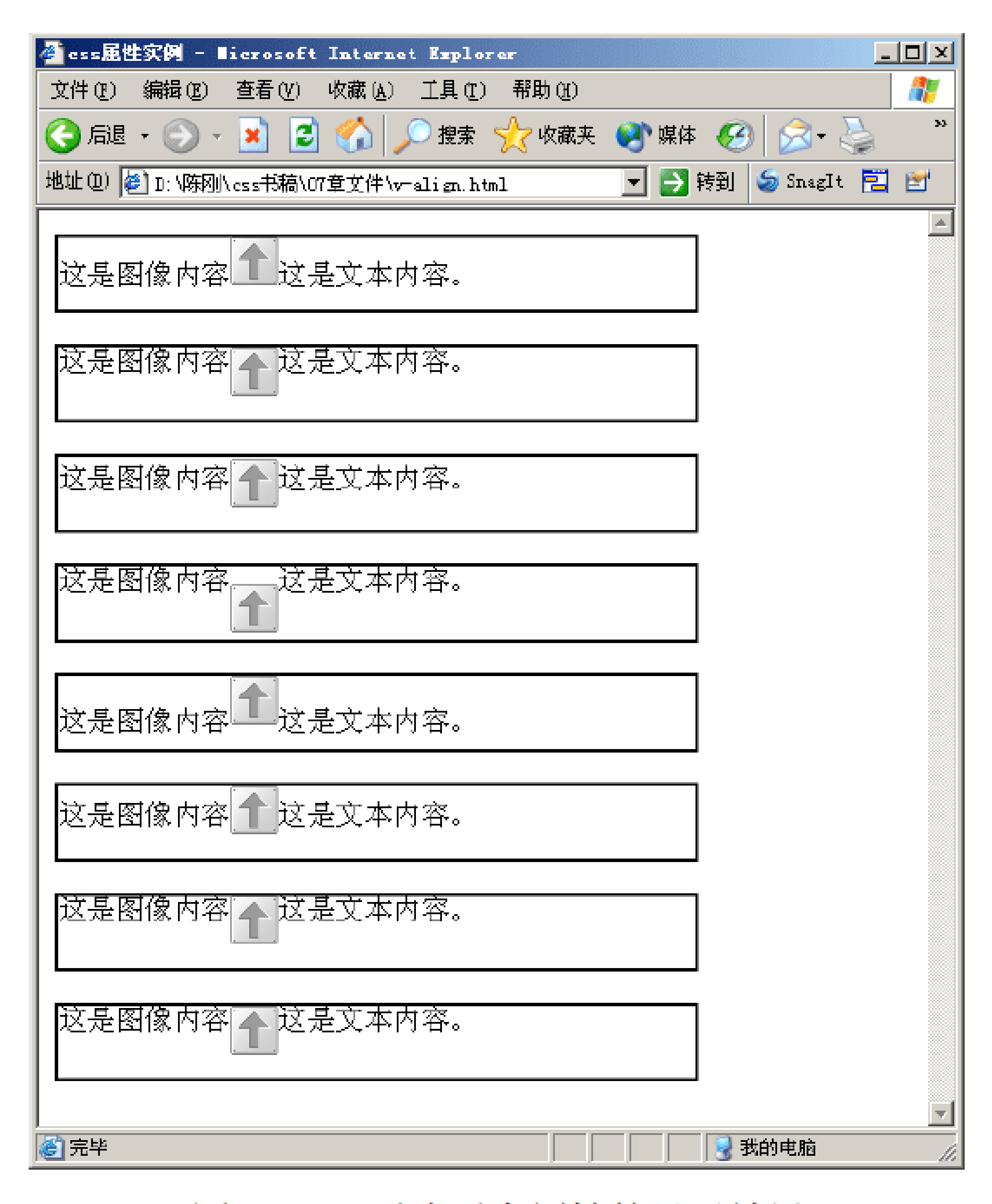


图 13-18　垂直对齐属性的显示效果

从图 13-18 可以看出，在垂直属性的各个属性值中，部分属性值的显示效果相同。

13.2.6 文本流向属性 layout-flow

文本流向属性(layout-flow)用来定义元素中文本流的显示方式。在文本流向属性中，可以使用两个属性值 horizontal 和 vertical-ideographic，其语法结构如下所示。

```
layout-flow : horizontal | vertical-ideographic;
```

其中每个属性的含义如下所示。

- horizontal：文本按照从左至右，然后从上到下的方式显示。
- vertical-ideographic：文本按照从上至下，然后从右到左的方式显示。

下面是一个使用文本方向属性的实例，其代码如下所示。

例程 13-19 example.html

```
01 <html xmlns="http://www.w3.org/1999/xhtml">
02 <head>
03 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
04 <title>文本的流向属性</title>
05 <style>
06 .p1
   {
   layout-flow:horizontal;
   }
07 .p2
   {
   layout-flow:vertical-ideographic;
   }
08 p
   {
   font-size:24px;
   width:400px;
   height:120px;
   border:4px solid yellow;
   }
09 </style>
10 </head>
11 <body>
   <p class="p1">这是一个显示方向属性的实例，注意这段话中文本的流向。</p>
   <p class="p2">这是一本显示方向属性的实例，注意这段话中文本的流向。</p>
12 </body>
13 </html>
```

以上的代码中，06 和 07 行中在两个<p>元素中分别定义了不同的文本流向属性值。其中第 1 个<p>元素中，定义文本从左至右显示。第 2 个<p>元素中，定义文本从上至下显示。同时定义了元素的宽度、高度、边框等属性，目的是更好地显示文本的流向和位置。代码运行后，其显示效果如图 13-19 所示。

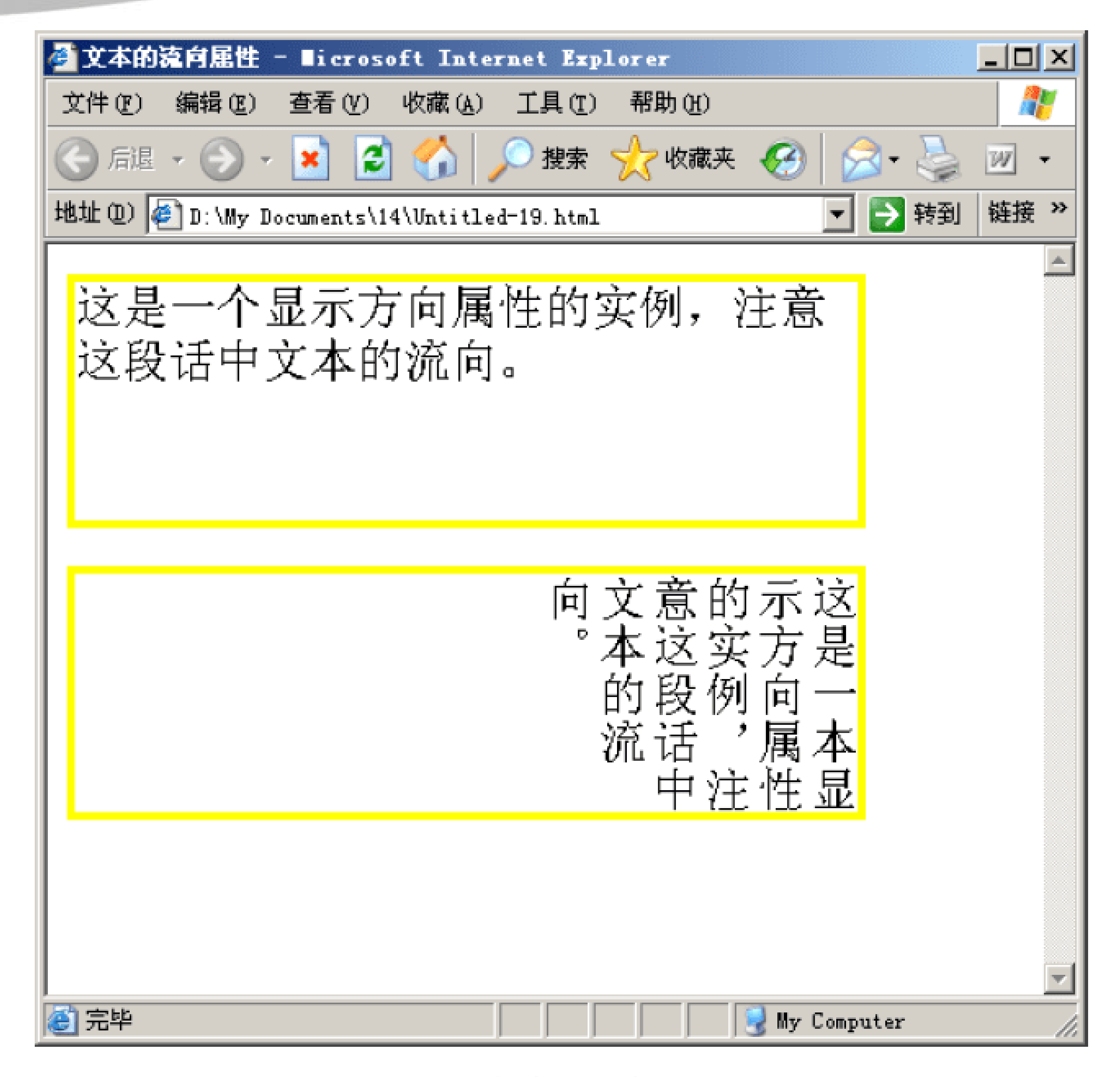


图 13-19　文本流向属性的显示效果

13.2.7　文本方向属性 direction

文本方向属性(direction)用来定义元素中文本的显示方向。在文本方向属性中，可以使用两个属性值 ltr 和 rtl，其语法结构如下所示。

```
direction : ltr | rtl;
```

其中每个属性的含义如下所示。

- ltr：文本按照从左至右的方向显示。
- rtl：文本按照从右到左的方向显示。



注意

使用文本方向属性只能改变文本的显示位置，并不能改变文本内容的流向。

下面是一个使用文本方向属性的实例，其代码如下所示。

例程 13-20　example.html

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>文本方向属性实例</title>
<style>
p
{
```

```
        font-size:36px;
        width:400px;
        height:100px;
        border:2px solid #000000;
    }
07  .p1
    {
    direction:ltr;
    }
08  .p2
    {
    direction:rtl;
    }
09  </style>
10  </head>
11  <body>
    <p class="p1">这段文本的方向是从左向右显示的</p>
    <p class="p2">这段文本的方向是从右向左显示的</p>
12  </body>
13  </html>
```

以上的代码中，07 和 08 行中在两个<p>元素中分别定义了不同的文本方向属性值。其中第 1 个<p>元素中，定义文本从左至右显示。第 2 个<p>元素中，定义文本从右至左显示。同时 06 行中定义了元素的宽度、高度、边框等属性，目的是更好地显示文本的方向和位置。代码运行后，其显示效果如图 13-20 所示，注意换行地方的显示形式。

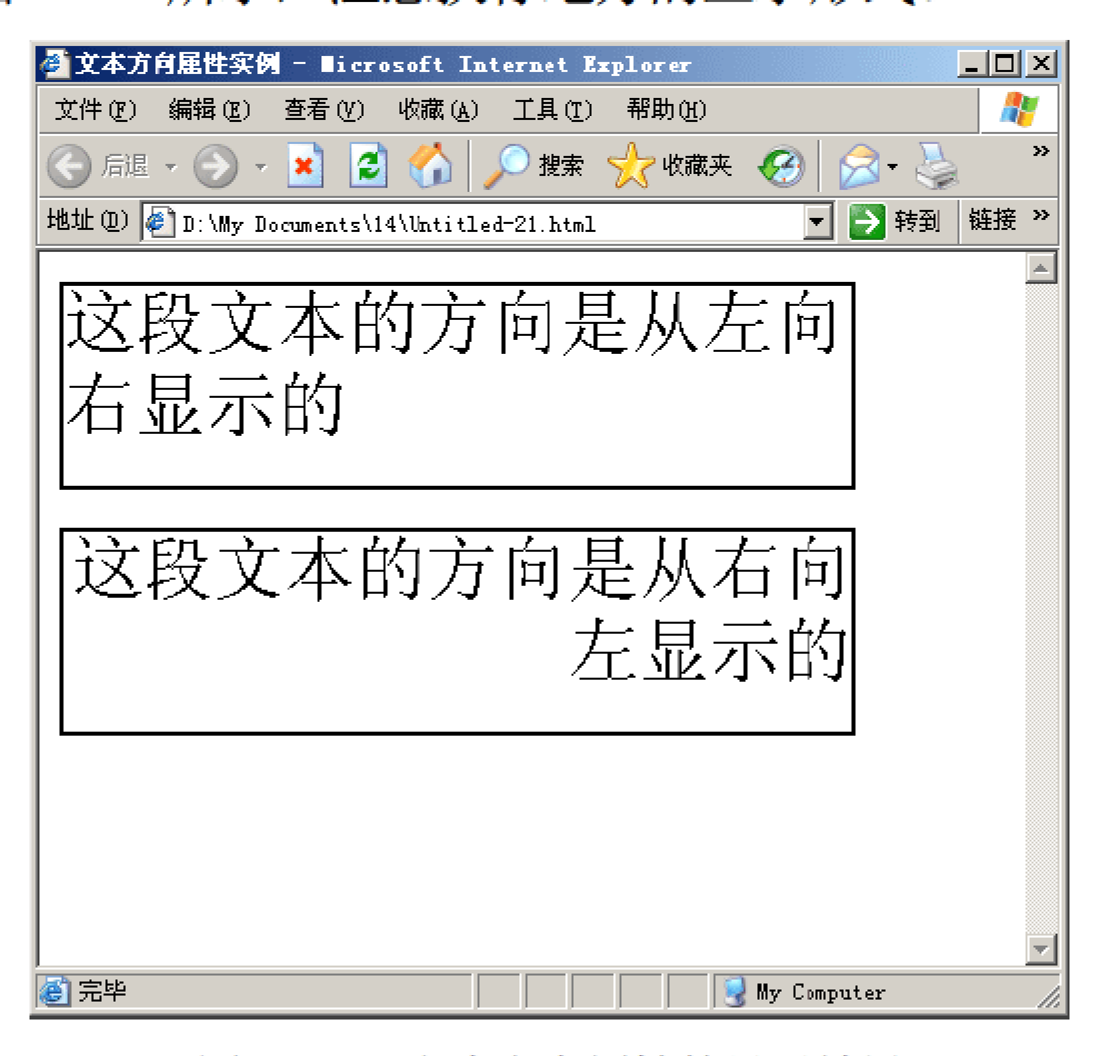


图 13-20　文本方向属性的显示效果

13.2.8　文本排序属性 unicode-bidi

文本排序属性(Unicode-bidi)用来定义文本内容的显示顺序。在文本排序属性中，可以使用 3 个属性值，分别为 normal、bidi-override 和 embed。其语法结构如下所示。

```
unicode-bidi : normal | bidi-override | embed
```

其中每个属性值的含义如下所示。

- nomal：定义文本按照默认的顺序显示。
- bidi-override：定义文本按照文本方向属性中定义的方向显示。
- embed：按照文本方向属性，在对象内部进行隐式重排序。

下面是一个使用文本排序属性的实例，其代码如下所示。

例程 13-21 example.html

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>文本排序属性实例</title>
<style>
p
{
  width:450px;
  height:40px;
  border:2px solid #CC00CC;
}
.p1
{
direction:ltr;
unicode-bidi:bidi-override;
}
.p2
{
  direction:ltr;
  unicode-bidi:embed;
}
.p3
{
  direction:rtl;
  unicode-bidi:bidi-override;
}
.p4
{
  direction:rtl;
  unicode-bidi:embed;
}
</style>
</head>
<body>
<p class="p1">注意这个文本的排序方式。</p>
<p class="p2">注意这个文本的排序方式。</p>
<p class="p3">注意这个文本的排序方式。</p>
<p class="p4">注意这个文本的排序方式。</p>
</body>
</html>
```

以上的代码中，06 行中定义了元素的宽度、高度、边框等属性，目的是更好地显示文本的方向和位置。07 到 10 行中在 4 个<p>元素中，分别定义了不同的文本方向属性和文本排序属性。其中第 1 个和第 2 个<p>元素中，定义文本方向为由左至右，并定义了不同的文本排序属性。在后面两个<p>元素中，定义了类似的属性，只是文本方向属性值为由右至左。代码运行后，其显示效果如图 13-21 所示。

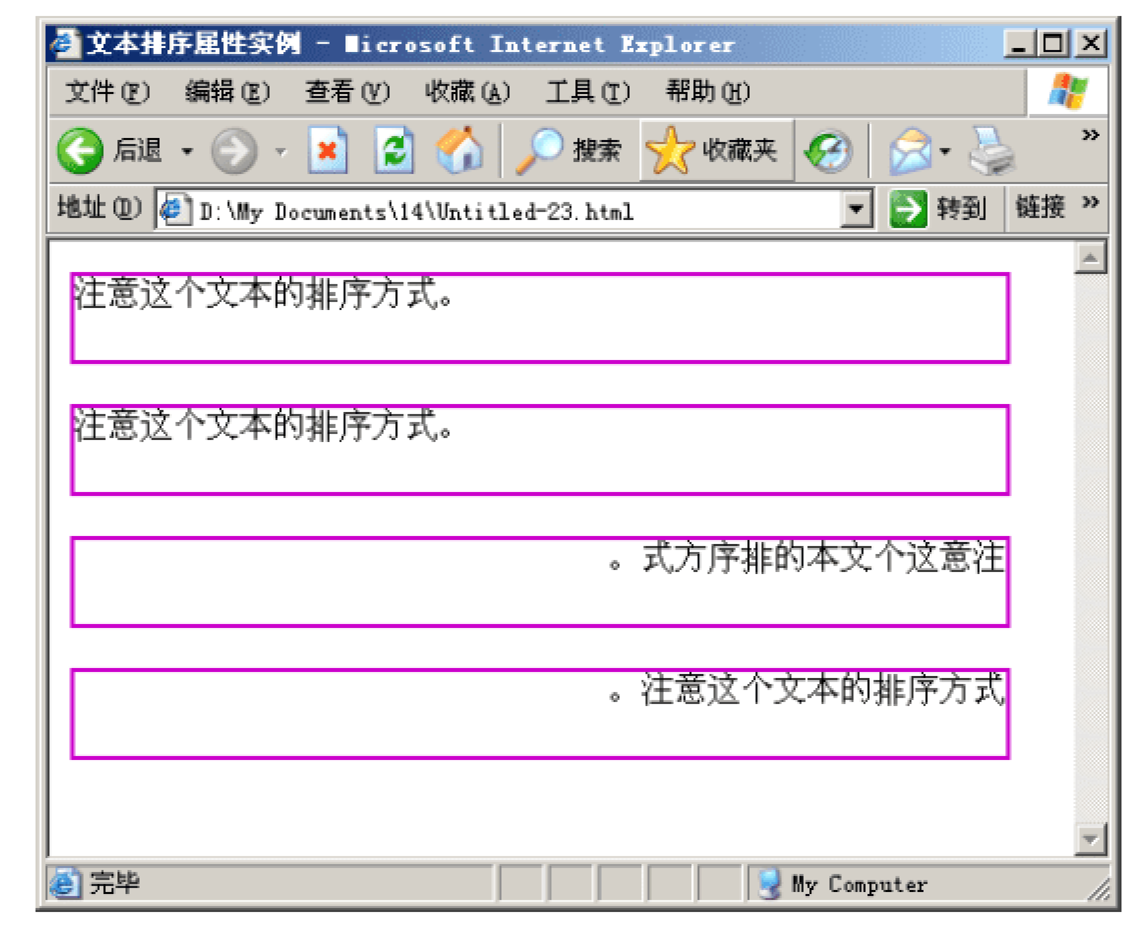


图 13-21 文本排序属性的显示效果

13.2.9 单词换行属性 word-break

单词换行属性(word-break)用来定义文本中能否在单词内部换行显示。在单词换行属性中，可以使用 3 个属性值，分别为 normal、break-all、keep-all。其语法结构如下所示。

```
word-break : normal | break-all | keep-all;
```

其中每个属性值的含义如下所示。

- nomal：定义使用默认的换行效果。
- break-all：定义文本中可以在单词内部换行显示。
- keep-all：定义文本单词或者文本不能换行显示。

下面是一个使用单词换行属性的实例，其代码如下所示。

例程 13-22 example.html

```
01  <html xmlns="http://www.w3.org/1999/xhtml">
02  <head>
03  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
04  <title>css 属性实例</title>
05  <style>
06  .p1
    {
     word-break:normal;
    }
07  .p2
    {
     word-break:break-all;
    }
08  .p3
    {
     word-break:keep-all;
    }
09  p{
     width:300px;
```

```
        height:50px;
        border:2px solid #000000;
      }
10    </style>
11    </head>
12    <body>
      <p class="p1">这是一个关于单词换行属性的实例 Howareyou。</p>
      <p class="p2">这是一个关于单词换行属性的实例 Howareyou。</p>
      <p class="p3">这是一个关于单词换行属性的实例 Howareyou。</p>
13    </body>
14    </html>
```

以上的代码中，06 到 08 行在 3 个<p>元素中，分别定义了不同的单词换行属性值。其中第 1 个<p>元素中，定义单词默认换行显示。第 2 个<p>元素中，定义单词可以从内部断开换行显示。第 3 个<p>元素中，定义单词不能换行显示。代码运行后，其显示效果如图 13-22 所示。

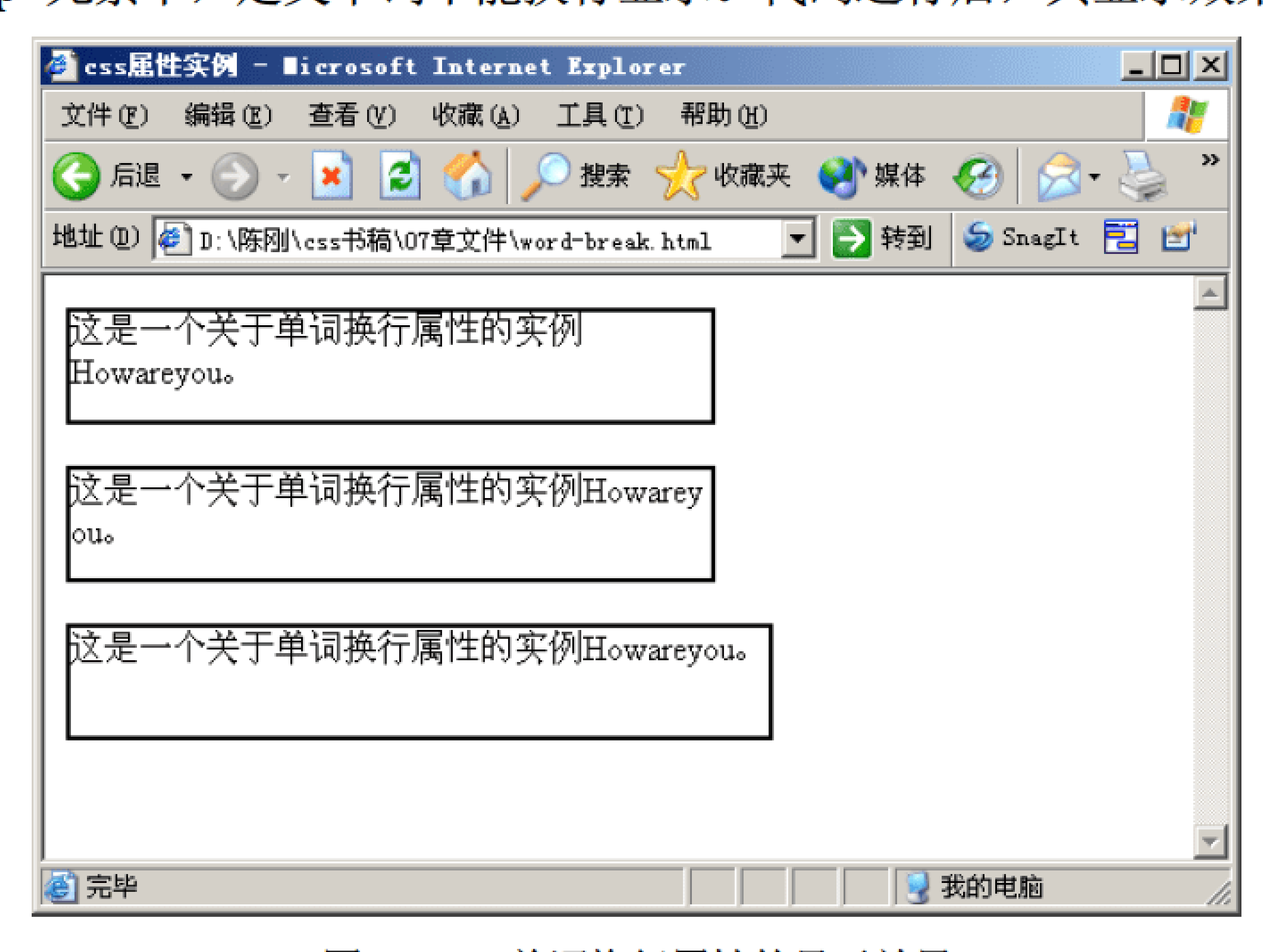


图 13-22　单词换行属性的显示效果

13.2.10　文本断开换行属性 word-wrap

文本断开换行属性(word-wrap)用来定义文本在容器中能否断开换行显示。在文本换行属性中，可以使用两个属性值，分别为 normal 和 break-word。其语法结构如下所示。

```
word-wrap : normal | break-word;
```

其中每个属性值的含义如下所示。

- nomal：定义文本同行显示。
- break-word：定义文本可以断开换行显示，同时允许文本可以在单词内换行显示。

下面是一个使用文本换行属性的实例，其代码如下所示。

例程 13-23 example.html

```
01  <html xmlns="http://www.w3.org/1999/xhtml">
02  <head>
03  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
04  <title>css 属性实例</title>
05  <style>
06  .p1
    {
     word-wrap:break-word;
    }
07  .p2
    {
     word-wrap:normal;
     word-break:normal;
    }
08  p{
     width:300px;
     height:50px;
     border:2px solid #000000;
    }
09  </style>
10  </head>
11  <body>
    <p class="p1">HowareyouHowareyouHowareyouHowareyouHowareyouHowareyouHowareyou。</p>
    <p class="p2">HowareyouHowareyouHowareyouHowareyouHowareyouHowareyouHowareyou。</p>
12  </body>
13  </html>
```

以上的代码中，06 行和 07 行在两个<p>元素中分别定义了不同的断开换行属性值。同时在元素中使用了一个比较特殊的文本内容，目的是显示断开的效果。代码运行后，其显示效果如图 13-23 所示。

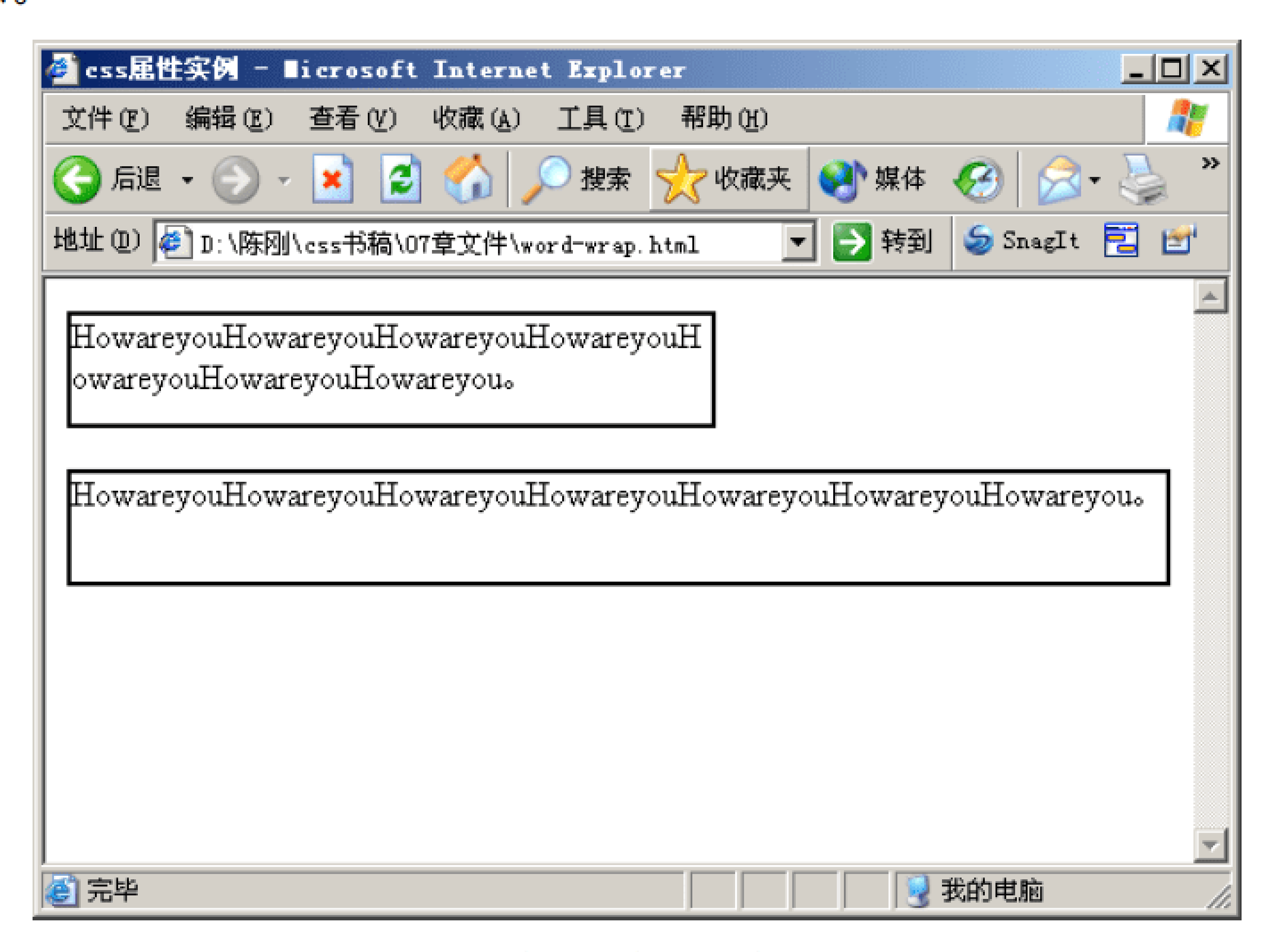


图 13-23　文本断开换行属性的显示效果

注意

在以上的实例中，使用了特殊的文本内容。如果使用普通的文本内容，换行和断开的效果将会以不同的方式显示。

下面将以上实例中使用的特殊文本内容换成较为普通的文本内容。使用相同的 CSS 样式，其显示效果如图 13-24 所示。

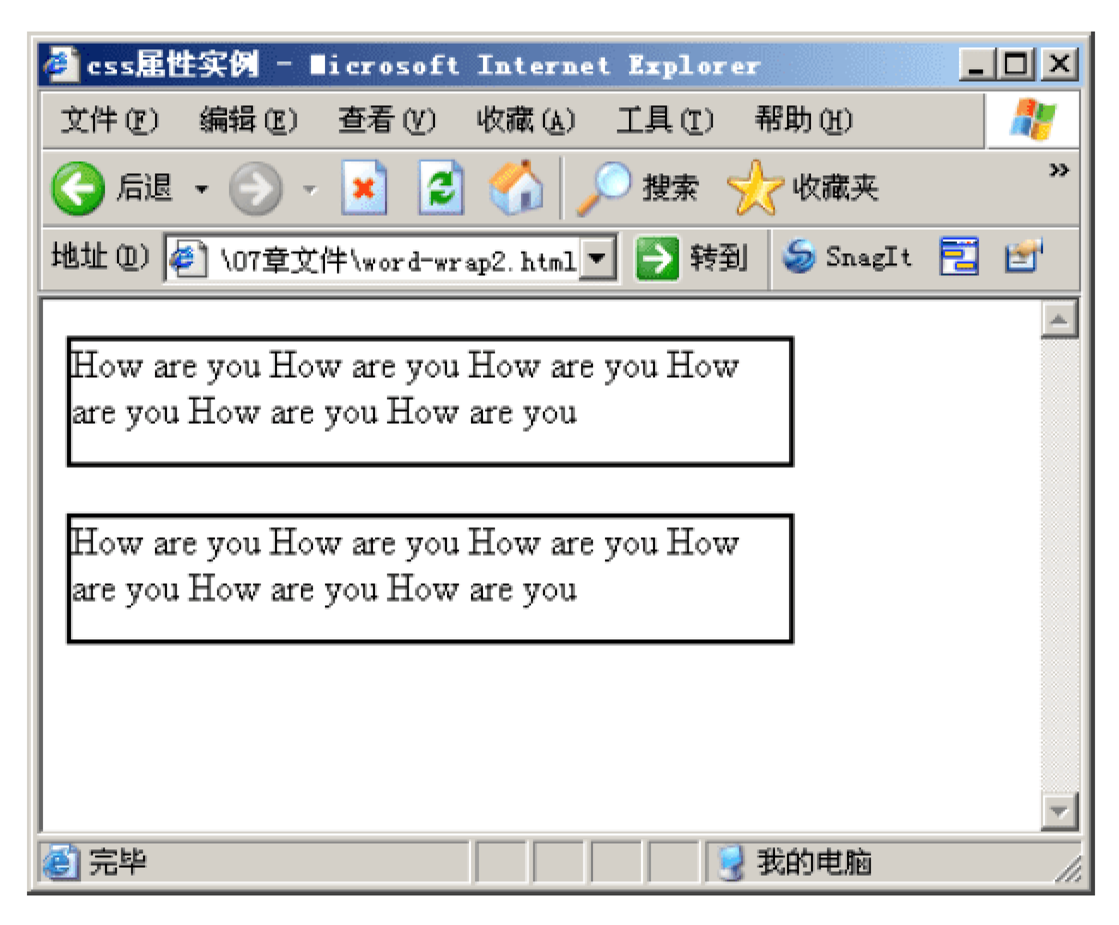


图 13-24　使用普通文本的断开换行效果

13.3 本章习题

一、选择题

1．若要以加粗宋体、12 号字显示“vbscript”，以下用法中，正确的是(　　)。

A.vbscript

B.vbscript

C.vbscript

D.vbscript

2．以下关于 FONT 标记符的说法中，错误的是：(　　)。

A. 可以使用 color 属性指定文字颜色。

B. 可以使用 size 属性指定文字大小(也就是字号)。

C. 指定字号时可以使用 1~7 的数字。

D. 语句 <FONT size="+2">这里是 2 号字</FONT> 将使文字以 2 号字显示。

3．以下有关样式表项的定义中，正确的是：(　　)。

A. H1{font-family:楷体_gb2312, text-aligh:center}

B. H1{font-family=楷体_gb2312, text-aligh=center}

C. H1{font-family:楷体_gb2312; text-aligh:center}

D. H1{font-family=楷体_gb2312; text-aligh=center}

4．以下有关样式表项的定义中，正确的是：(　　)。

A. P{font-size=24px, text-aligh=center}

B. P{font-size:24px, text-aligh:center}

C. P{font-size=24px; text-aligh=center}

D. P{font-size: 24px; text-aligh:center}

5．letter-spacing 表示的意思是(　　)。

A. 单词间隔属性　B. 字母间隔属性　C. 字体修饰属性　D. 字母间隔属性

6．字体(font)样式的属性不包括(　　)。

A. font-family　B. font-style　C. font-weight　D. font-Italic

二、填空题

1．在 CSS 中，字体的控制包括控制文本的________、字体的大小、__________、字体的颜色、__________等方面的内容。

2．字体样式属性(font-style)用来定义字体的显示样式。在字体样式属性中，可以使用的属性值包括_______、_______、________，分别表示_________、____________、__________。

3．文本空白属性(text-overflow)可以使用 3 个属性值，分别为_______、______、_______。

三、实战练习

制作一个加入 CSS 样式的页面，应用各种字体属性对页面文字进行处理。

Chapter 14

CSS控制列表元素的显示

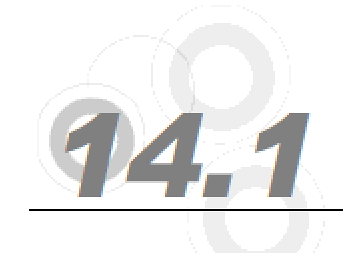
在网页中，列表元素通常用来定义导航，或者文章标题列表等内容。在 CSS 中，可以通过相应的属性控制列表元素的各种显示效果。在本章中，将对 CSS 中控制列表元素的显示和列表元素的使用与嵌套做详细的讲解，学习的时候可通过对照前面所学知识来加深对本章的理解。

本章主要内容有：

◎ 重点掌握 CSS 控制列表元素的使用方法。

◎ 熟悉列表元素的各种属性。

◎ 学会使用嵌套的列表。



14.1 控制列表元素的显示

在 CSS 中，列表元素的控制包括控制列表符号、列表图像、列表位置等几个方面的内容。通过定义各种属性可以更改列表的默认显示方式，但是想要完全控制列表元素的显示，还需要依赖其他的 CSS 属性。下面将一一进行详细讲解。

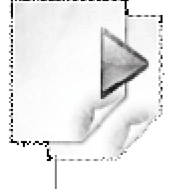
14.1.1 列表符号属性 list-style-type

列表符号属性(list-style-type)用来定义列表中使用的预设符号。其中使用的属性值有很多，部分属性值还不被浏览器所支持。其语法结构如下所示。

```
list-style-type : disc | circle | square | decimal | lower-roman | upper-roman | lower-alpha | upper-alpha | none | armenian | cjk-ideographic | georgian | lower-greek | hebrew | hiragana | hiragana-iroha | katakana | katakana-iroha | lower-latin | upper-latin;
```

其中部分属性值的含义如下所示。

- disc：实心圆点。
- circle：空心圆圈。
- square：实心方块。
- decimal：阿拉伯数字。
- lower-roman：小写罗马数字。
- upper-roman：大写罗马数字。
- lower-alpha：小写英文字母。
- upper-alpha：大写英文字母。
- none：不使用任何符号。



注意

在以上讲解的几个属性值以外的属性值，还不被大多数浏览器所支持。

下面是一个使用列表符号属性的实例，其代码如下所示。

例程 14-1 example.html

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title> 列表符号属性</title>
<style>
.liststyle1
{
 list-style-type:disc;}
```

```
.liststyle2
{
 list-style-type:circle;}
.liststyle3
{
 list-style-type:square;
}
.liststyle4
{
 list-style-type:decimal;
}
.liststyle5
{
 list-style-type:lower-roman;
}
.liststyle6
{
 list-style-type:upper-roman;
}
.liststyle7
{
 list-style-type:lower-alpha;
}
.liststyle8
   {
 list-style-type:upper-alpha;
}
.liststyle9
{
list-style-type:none;
}
</style>
</head>
<body>
<ul>
<li class="liststyle1">实心圆点</li>
<li class="liststyle2">空心圆圈</li>
<li class="liststyle3">实心方块</li></ul>
<ul class="liststyle4">
<li>阿拉伯数字</li>
<li>阿拉伯数字</li></ul>
<ul class="liststyle5">
<li>小写罗马字</li>
<li>小写罗马字</li></ul>
<ul class="liststyle6">
<li>大写罗马字</li>
<li>大写罗马字</li></ul>
<ul class="liststyle7">
<li>小写英文字母</li>
<li>小写英文字母</li></ul>
```

```
24  <ul class="liststyle8">
    <li>大写英文字母</li>
    <li>大写英文字母</li></ul>
25  <ul class="liststyle9">
    <li>不用任何符号</li></ul>
26  </body>
27  </html>
```

以上的代码中，06 行～15 行中分别定义了 9 种使用不同列表符号属性值的样式。其中，由于某些属性值会涉及到编号的问题，所以使用了两个<li>元素来显示。代码运行后的显示效果如图 14-1 所示。

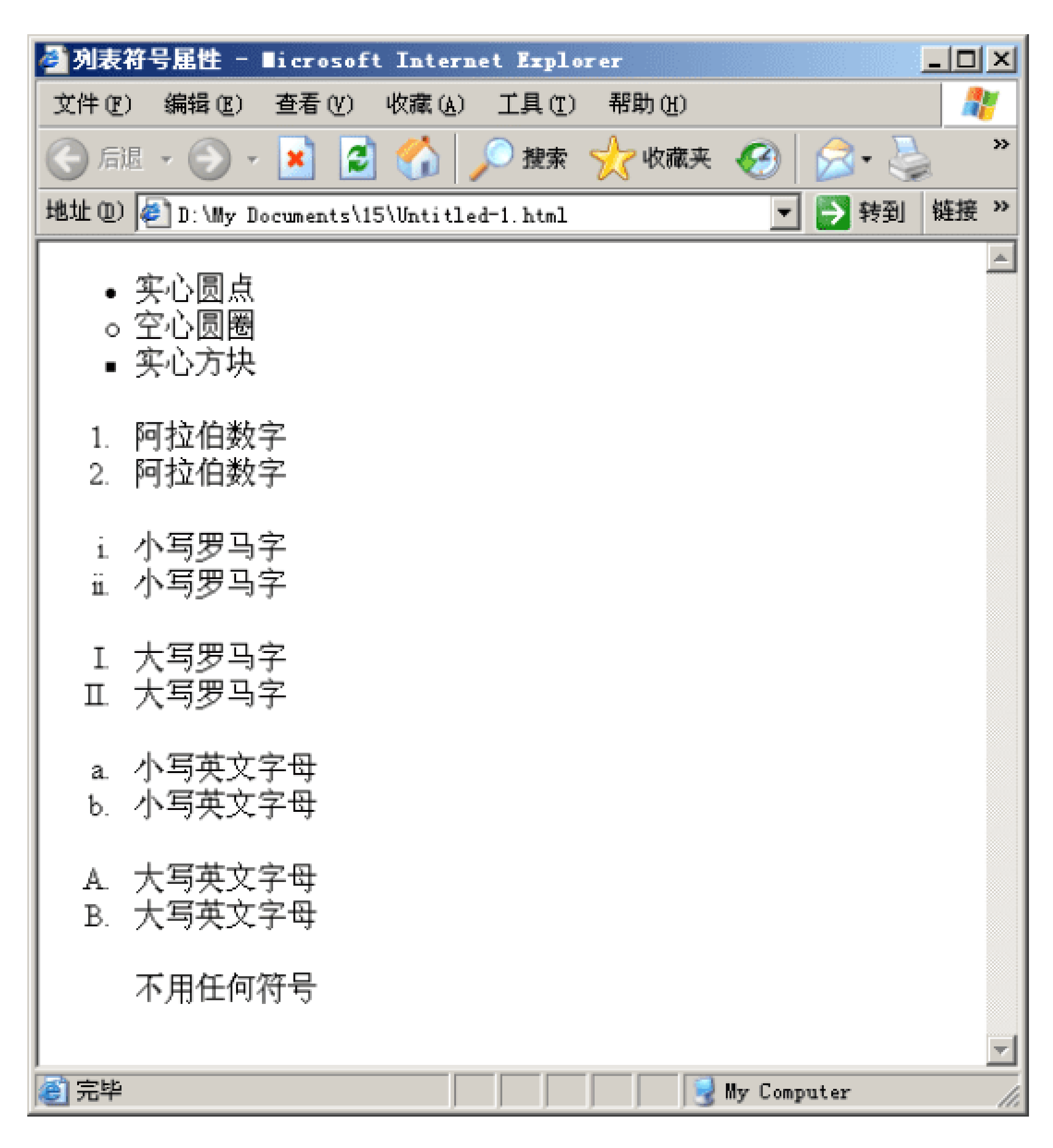


图 14-1　列表符号属性的显示效果

14.1.2　列表符号的混用

在定义列表元素的时候，有时候会混用各种列表符号。当混用的列表符号中包含顺序问题的时候，同一列表中将会计算所有列表项目的数目，确定当前列表项目的显示方式。

下面是一个混用列表符号的实例，其代码如下所示。

例程 14-2　example.html

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>列表符号的混用</title>
<style>
.liststyle1
{
 list-style-type:disc;
```

```
    }
07  .liststyle2
    {
     list-style-type:circle;
    }
08  .liststyle3
    {
     list-style-type:upper-alpha;
     }
09  .liststyle4
    {
    list-style-type:lower-roman;
    }
10  li
    {
    font-size:24px;
    }
11  </style>
12  </head>
13  <body>
    <ul>
    <li class="liststyle1">注意显示的列表项目</li>
    <li class="liststyle2">注意显示的列表项目</li>
    <li class="liststyle3">注意显示的列表项目</li>
    <li class="liststyle4">注意显示的列表项目</li></ul>
14  </body>
15  </html>
```

以上的代码中，定义不同了的列表元素属性，同时定义了<li>元素中文本的大小，目的是更好地显示列表项目的效果。以上代码的显示效果，如图 14-2 所示。

从图 14-2 可以看出，在混用的列表中，虽然第一次使用大写英文字母属性值(decimal)，但是由于当前文本的位置为第 3 个，所以最终会显示大写英文字母 C。定义其他列表符号，也会出现相同的效果。

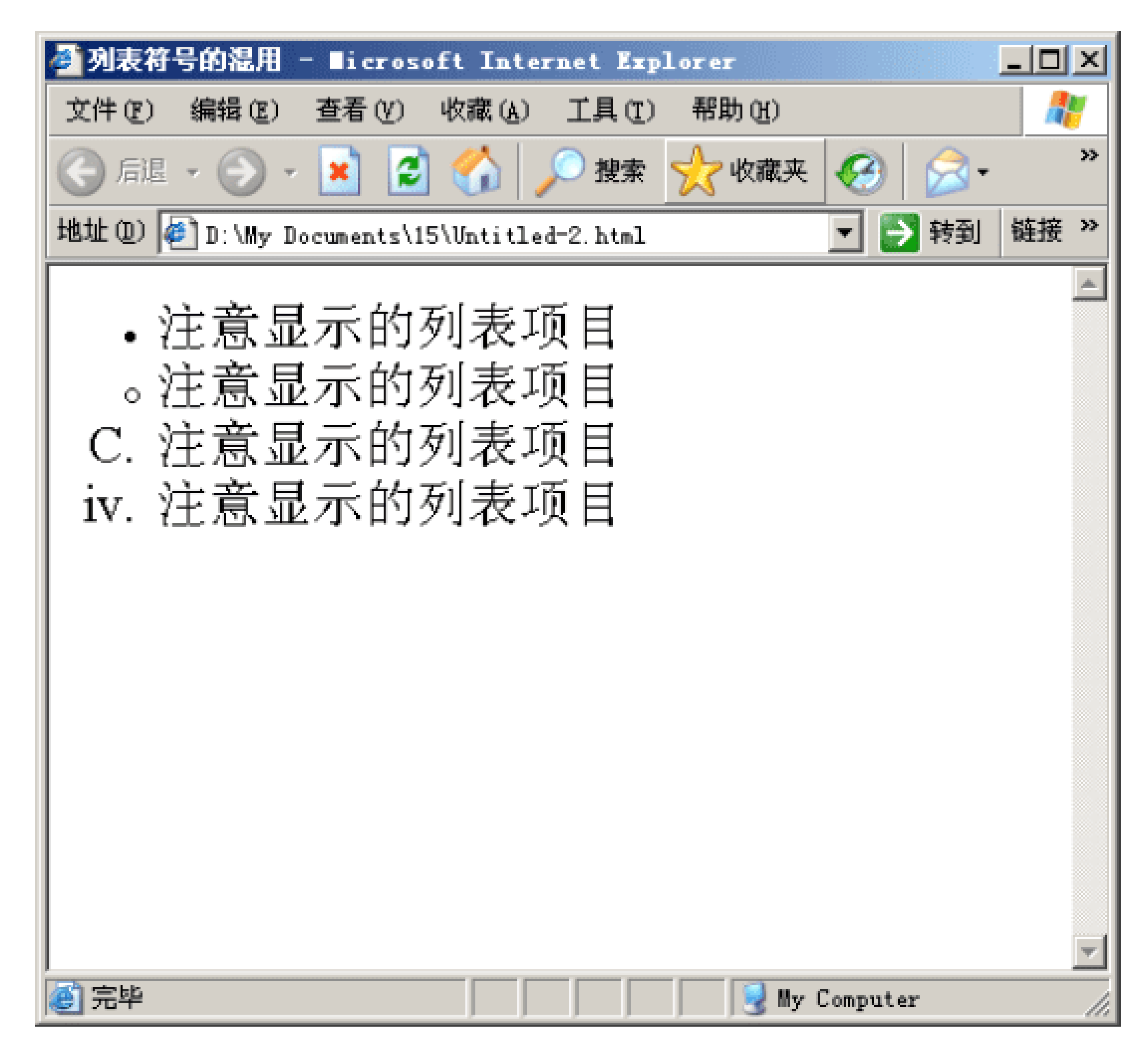


图 14-2　混用列表符号的显示效果

14.1.3　列表图像属性 list-style-image

列表图像属性(list-style-image)用来定义列表元素中替换列表符号的图像。在列表图像属性中，可以使用两个属性值：none 和 url，其语法结构如下所示。

```
list-style-image : none | url;
```

其中各个属性值的含义，如下所示。

- none：不使用任何列表图像。
- url：定义使用列表图像的路径。

下面是一个使用列表图像属性的实例，其代码如下所示。

例程 14-3　example.html

```
01  <html xmlns="http://www.w3.org/1999/xhtml">
02  <head>
03  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
04  <title>列表图像属性属性实例</title>
05  <style>
06  li
    {
    list-style-image:url(pic.jpg);
    font-size:24px;
    }
07  </style>
08  </head>
09  <body>
    <ul>
    <li>使用列表图像的路径</li>
    <li>使用列表图像的路径</li>
    <li>使用列表图像的路径</li></ul>
10  </body>
11  </html>
```

以上的代码中，06 行在<li>元素里定义了列表图像的路径。此时所有的<li>元素都将使用定义的列表图像。以上代码的显示效果如图 14-3 所示。

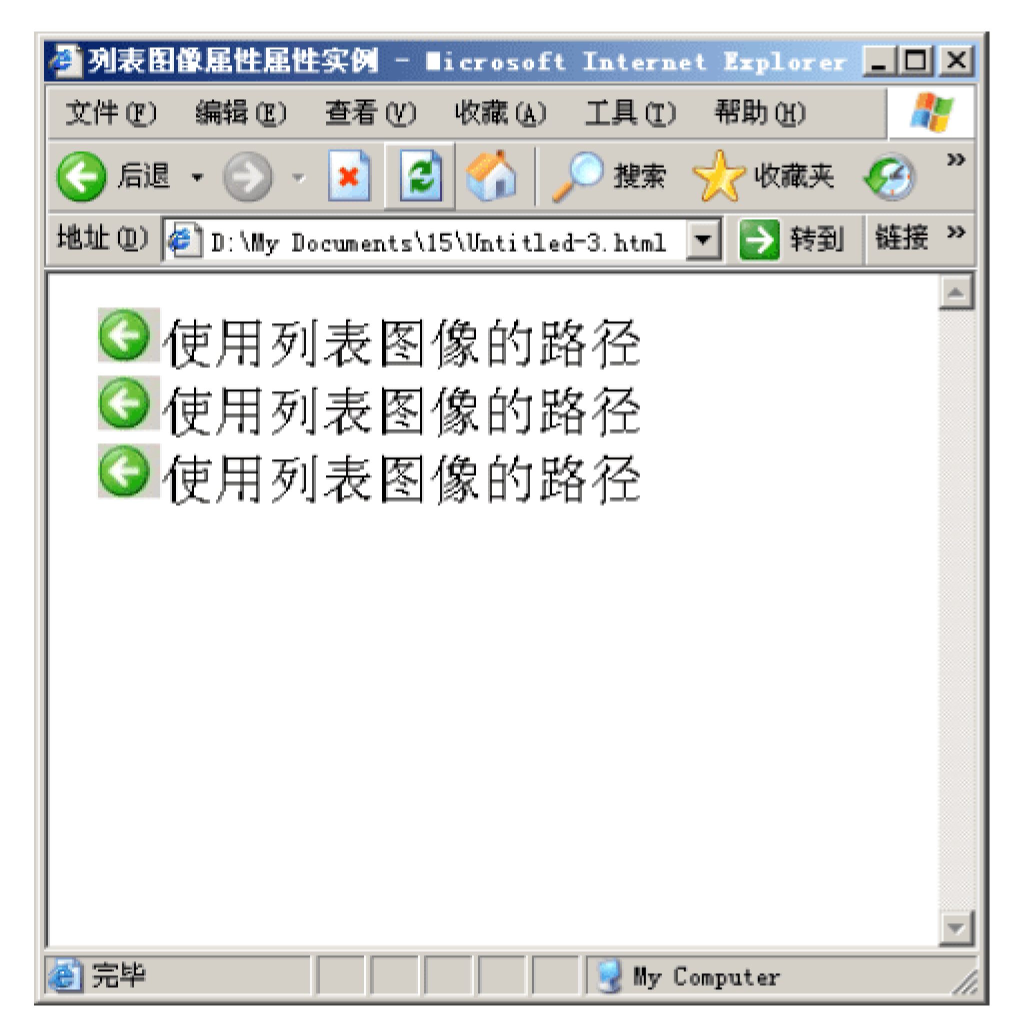


图 14-3　列表图像属性的显示效果

14.1.4　列表图像的显示位置

在列表图像属性中，使用的列表图像的显示位置将与文本的底部基线对齐。所以在使用列表图像的时候，要注意选择合适的图像大小，否则将会显示异常。

下面是一个显示列表图像位置的实例，其代码如下所示。

例程 14-4　example.html

```
01  <html xmlns="http://www.w3.org/1999/xhtml">
02  <head>
03  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
04  <title>列表图像属性属性实例</title>
05  <style>
06  li
    {
    list-style-image:url(pic1.jpg);
    font-size:18px;
    }
07  </style>
08  </head>
09  <body>
    <ul>
    <li>列表图像和字体</li>
    <li>列表图像和字体</li>
    <li>列表图像和字体</li></ul>
10  </body>
11  </html>
```

以上的代码中，06 行定义了一个较大的列表图像，同时定义了一个较小的字体大小。以上代码的显示效果如图 14-4 所示。

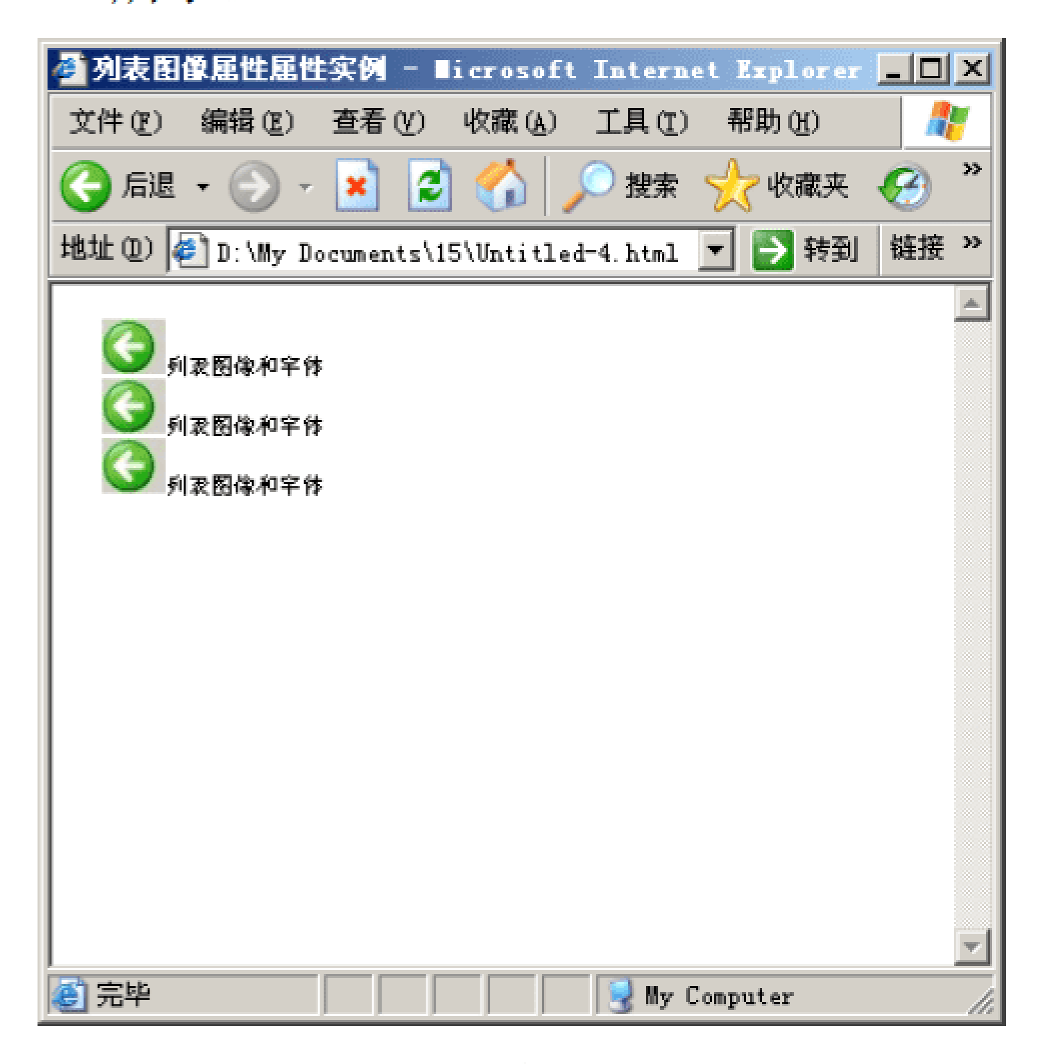


图 14-4　列表图像位置的显示效果

14.1.5 标记位置属性 list-style-position

标记位置属性(list-style-position)用来定义列表中标记的显示位置。在字体样式属性中，可以使用两个属性值：outside 和 inside。其语法结构如下所示。

```
list-style-position : outside | inside;
```

其中各个属性值的含义如下所示。

- outside：定义列表标记显示在文本之外。
- inside：定义列表标记显示在文本之内。

下面是一个使用标记位置属性的实例，其代码如下所示。

例程 14-5 example.html

```
01  <html xmlns="http://www.w3.org/1999/xhtml">
02  <head>
03  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
04  <title> 标记位置属性属性实例</title>
05  <style>
06  li{
    list-style-type:square;
    font-size:24px;
    border:2px solid yellow
    }
07  .li1
    {
    list-style-position:inside;
    }
08  .li2
    {
    list-style-position:outside;
    }
09  </style>
10  </head>
11  <body>
    <ul class="li1">
    <li>这是一个使用标记显示在文本之内位置属性的实例，注意换行后标记的显示位置。</li>
    </ul>
    <ul class="li2">
    <li>这是一个使用标记显示在文本之外位置属性的实例，注意换行后标记的显示位置。</li>
    </ul>
12  </body>
13  </html>
```

以上的代码中，07 和 08 行分别在两个 li 元素中，定义了不同的标记位置属性值。同时定义列表符号为实心方块，字体大小为 24px，以便标记位置属性值的效果能够更加明显。以上代码的显示效果，如图 14-5 所示。

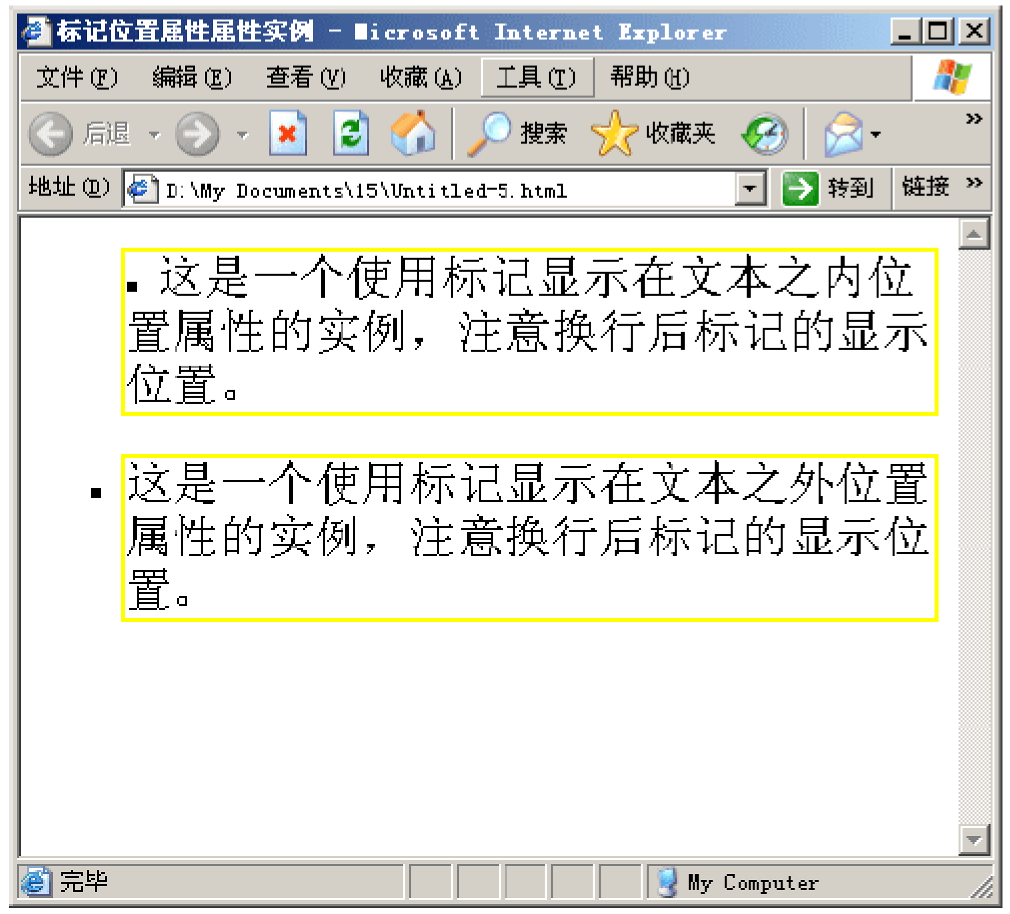


图 14-5　符号位置属性的显示效果

从图 14-5 可以看出，当定义的标记位置属性值为 inside 时，当换行时，列表标记显示在文本的内部。当定义的标记位置属性值为 outside 时，当换行时，列表标记显示在文本的外部。同时列表标记显示在列表内容的第一行底部。

14.1.6　标记位置属性与列表高度

当使用列表标记，同时定义了列表高度时，标记的显示位置、列表高度和标记位置有关。列表标记会显示在列表定义高度的底部，而不会显示在列表内容第一行之中。

下面是一个标记位置属性与列表高度的实例，其代码如下所示。

例程 14-6　example.html

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>css 属性实例</title>
<style>
.li1
{
list-style-position:inside;
}
.li2
{
list-style-position:outside;
}
li{
list-style-type:circle;
height:70px;
border:2px solid #000000;
font-size:24px;
}
</style>
```

```
10  </head>
11  <body>
    <ul class="li1">
    <li>这是一个使用标记位置属性的实例，注意换行后标记的显示位置。</li>
    <li>这是一个使用标记位置属性的实例，注意换行后标记的显示位置。</li></ul>
    <ul class="li2">
    <li>这是一个使用标记位置属性的实例，注意换行后标记的显示位置。</li>
    <li>这是一个使用标记位置属性的实例，注意换行后标记的显示位置。</li></ul>
12  </body>
13  </html>
```

以上的代码中，06 行和 07 行在两个 li 元素中定义了不同的标记位置属性值。同时 08 行定义了内容的高度、边框的大小、字体大小等各种表现属性，目的是使列表项目的显示效果更加明显。以上代码运行后，可以看到实心方块显示在框架之外，但是所在的位置不同。代码运行的显示效果如图 14-6 所示。

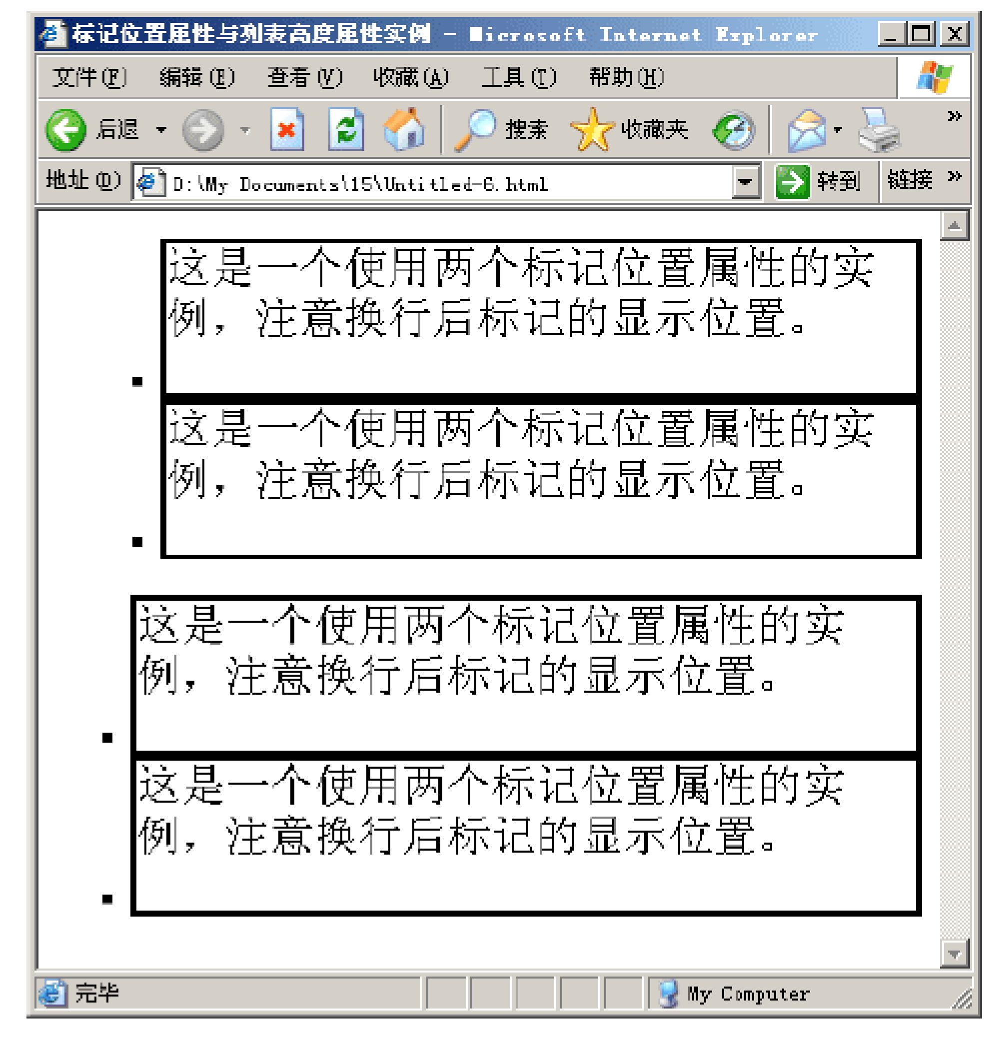


图 14-6　标记位置属性与列表高度的显示效果

14.1.7　列表综合属性 list-style

列表综合属性(list-style)用来统一定义列表的各种显示效果。在列表综合属性中，可以同时定义列表的标记位置、使用图片、列表符号等属性。其语法结构如下所示。

```
list-style : list-style-image || list-style-position || list-style-type;
```

在使用列表综合属性的时候，交换列表各种属性值的位置，并不影响属性的显示效果。

注意

在使用列表综合属性的时候，如果同时定义了列表的图片属性和列表符号属性，则会显示列表图片，而忽略定义的列表符号。

下面是一个使用列表综合属性的实例，其代码如下所示。

例程 14-7　example.html

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>列表综合属性实例</title>
<style>
li
{
```

```
        list-style:url(pic.jpg) inside disc;
        font-size:36px;
    }
07  </style>
08  </head>
09  <body>
    <ul>
    <li>注意这个使用列表综合属性的实例。</li>
    <li>注意这个使用列表综合属性的实例。</li></ul>
10  </body>
11  </html>
```

以上的代码中，06 行定义<li>元素的列表综合属性值中，列表图片路径为“url(pic.jpg)”，同时定义了文本大小为 36px，用来显示换行的效果。以上代码的显示效果如图 14-7 所示。

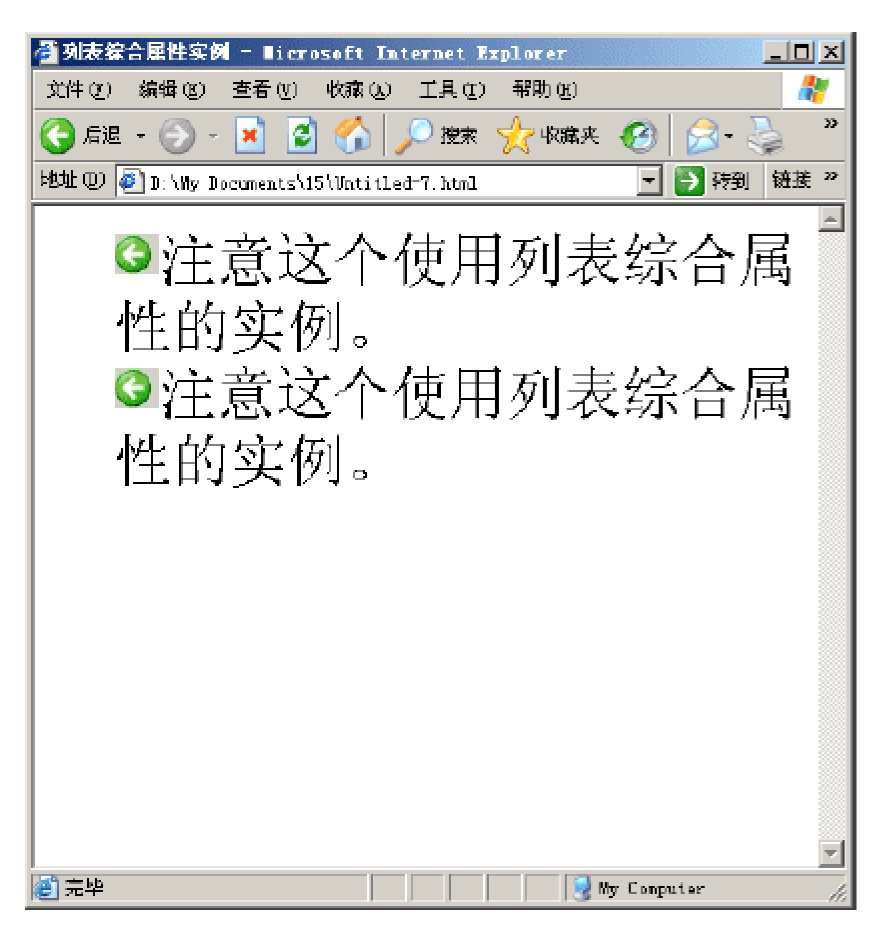


图 14-7　列表综合属性的显示效果

14.2 列表元素的使用和嵌套

在制作页面的时候，经常会使用列表元素来制作各种页面内容。在 HTML 中，列表元素默认会包含各种预先定义的显示效果，这些显示效果在不同的浏览器中使用的属性值也不相同。所以在使用 CSS 控制列表元素的时候，要对列表元素的显示效果有所了解。同时使用嵌套的列表元素可以制作出各种复杂的显示效果。具体内容如下所示。

14.2.1　列表元素的默认属性值

在列表元素中，如果未定义任何表现属性的时候。在 IE 浏览器中，会为列表定义默认的边界属性。在 Firefox 浏览器中，会为列表同时定义默认的补白和边界属性。

下面是一个关于列表元素默认显示效果的实例，其代码如下所示。

例程 14-8 example.html

```
01  <html xmlns="http://www.w3.org/1999/xhtml">
02  <head>
03  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
04  <title>列表元素的默认属性实例</title>
05  <style>
06  div
    {
    border:3px solid red;
    }
07  ul
    {
    list-style:url(pic.jpg)    outside circle;
    background:blue;
    font-size:24px;
    }
08  </style>
09  </head>
10  <body>
11  <div>
      <ul>
      <li>这里使用了列表元素，注意这句话的显示效果。</li>
      <li>这里使用了列表元素，注意这句话的显示效果。</li></ul>
12  </div>
13  </body>
14  </html>
```

以上的代码中，06 行在<div>元素中定义了边框属性。同时 07 行在列表元素中定义了列表的标记符号、列表的图像等属性，并定义了列表的背景，便于显示列表中各个部分占有的空间，以及边界等的显示。其显示效果如图 14-8 所示。

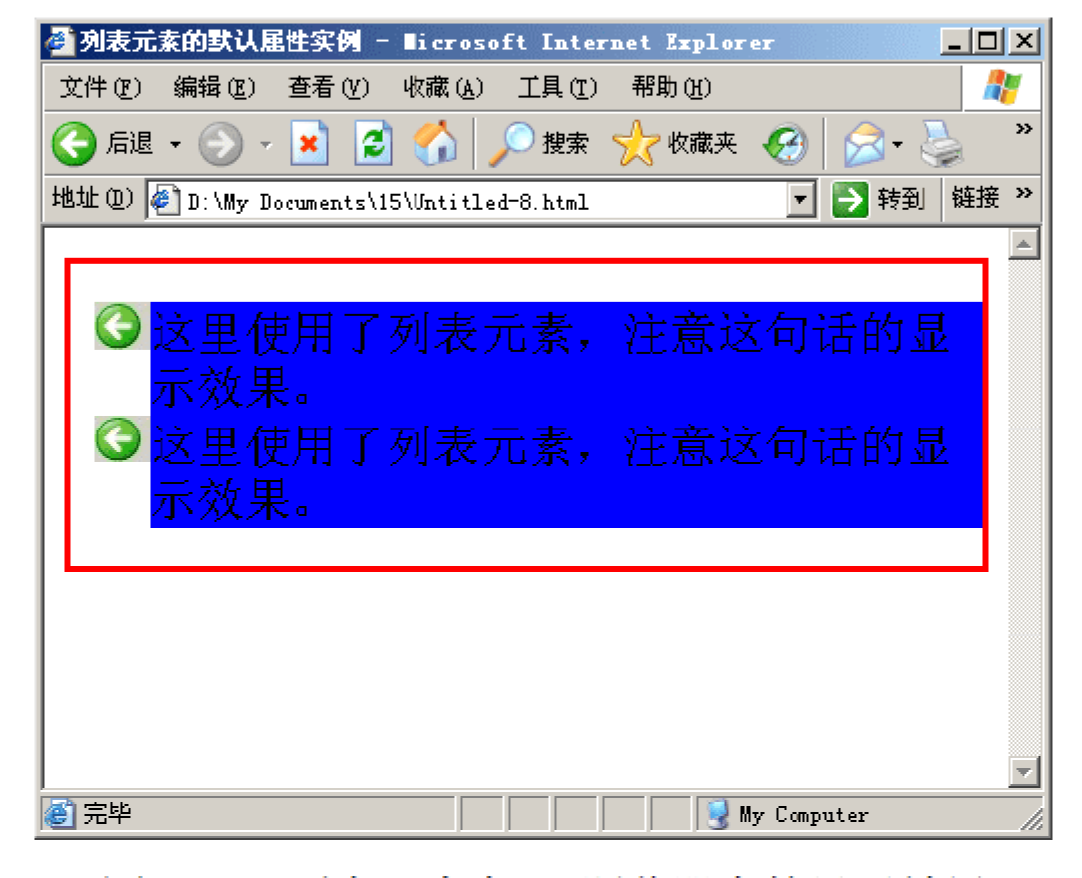


图 14-8　列表元素在 IE 浏览器中的显示效果

14.2.2　统一列表元素的边界和补白

通过 14.2.1 节的讲解，可以知道，造成列表元素显示差异的主要原因是：存在不同的边界和补白属性值。所以可以通过定义边界和补白属性，来统一列表元素的显示差异。

下面是一个统一列表元素显示效果的实例，其代码如下所示。

例程 14-9 example.html

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>无标题文档</title>
<style>
div
```

```
        {
        border:4px solid #000000;}
07      ul
        {
        margin:0;
        padding:0;
        list-style:inside disc;
        background:yellow;
        font-size:30px;}
08      </style>
09      </head>
10      <body>
11      <div>
          <ul>
          <li>列表元素显示效果的实例。</li>
          <li>列表元素显示效果的实例。</li></ul>
12      </div>
13      </body>
14      </html>
```

以上的代码中，07 行在<ul>元素中分别定义了边界和补白属性值为 0。同时定义了父元素和元素本身的各种显示属性，目的是用来显示补白和边界的范围。代码运行后在 IE 浏览器中的显示效果如图 14-9 所示。在 Firefox 浏览器中的显示效果如图 14-10 所示。



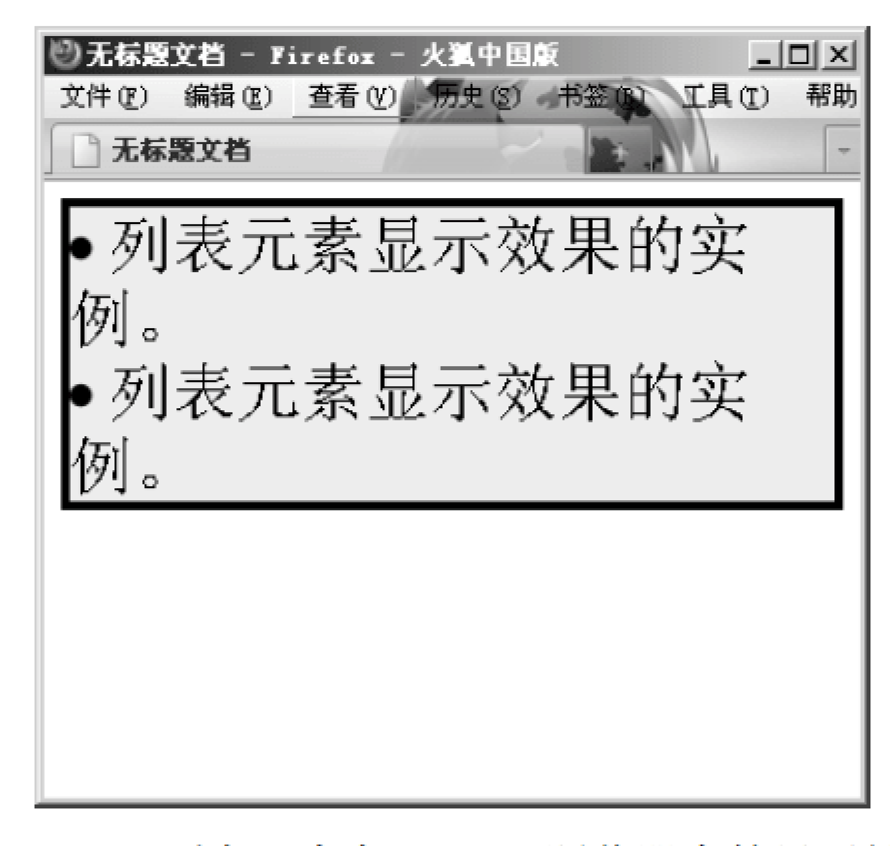
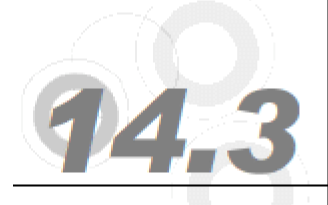


图 14-9　列表元素在 IE 浏览器中的显示效果　图 14-10　列表元素在 Firefox 浏览器中的显示效果

从图 14-10 可以看出，当定义了相同的边界和补白属性后，列表元素的显示效果在两个浏览器中基本相同了。



14.3 本章习题

一．选择题

1．以下那个列表属性值的作用是在页面中形成方块符号的？(　　)。

A. disc　B. circle　C. square　D. decimal

2．以下有关列表属性的说法中，正确的是：(　　)

A. 使用标记位置属性的属性值 outside 时，列表符号标记会显示在文本之内。

B. 如果同时定义了列表的图片属性和列表符号属性，则会显示列表图片，而忽略定义的列表符号。

C. 在混用的列表中，如果第三行第一次使用大写英文字母属性值，会显示大写英文字母 A。

D. 如果同时定义了列表的图片属性和列表符号属性，则会显示列表符号，而忽略定义的列表图片。

3．liststyle{list-style-type:upper-roman;} 表示的意思是(　　)。

A. 本列的列表符号为小写罗马字母

B. 本列的列表符号为大写罗马字母

C. 本列的列表符号为小写英文字母

D. 本列的列表符号为大写英文字母

4．关于列表综合属性(list-style)的使用不正确的是：(　　)。

A. 列表综合属性可以用来统一定义列表的各种显示效果

B. 列表综合属性同时定义列表的标记位置、使用图片、列表符号等属性

C. 使用列表综合属性的时候，交换列表各种属性值的位置，并不影响属性的显示效果。

D. 使用列表综合属性的时候，交换列表各种属性值的位置，就会影响属性的显示效果。

二、填空题

1．在页面中使用大写罗马字母作为列表符号的代码是__________________________。

2．列表图像属性(list-style-image)中，可以使用两个属性值______和_______，分别表示_______________________和___________________。

三、实战练习

1．制作一个 6 行混用列表符号的实例的页面。要求各行中依次使用方块、数字、大写英文字母、小写罗马字母和图片(图片的大小要合适)。最后一行默认。注意它们各自在页面中的效果。

2．制作一个使用列表符号的实例的页面，并统一列表元素的边界和补白为 50，要求定义列表标记显示在文本之外，背景为灰色，字体为黑色。

Chapter 15

CSS控制表格元素的显示

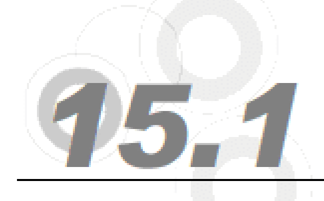
表格元素，一般用来显示矩阵格式的数据。在 CSS 中，可以通过相应的表现属性来定义表格元素的表现效果。除了可以定义表格的边框、补白、边界等属性之外，还可以使用表格独有的 CSS 属性，如边框合并属性(border-collapse)等，对表格进行进一步的控制。

本章主要内容有：

◎ 熟悉 CSS 控制表格元素边框的各个属性及其应用。

◎ 属性单元格的制约关系。

◎ 精确控制表格与单元格的大小。



15.1 控制表格元素的显示

在 CSS 中，表格元素的控制包括控制表格边框是否合并、控制表格边框间距、表格标题位置等几个方面的内容。由于浏览器对 CSS 属性的支持情况不同，部分属性只有在特定的浏览器中才能够显示相应的效果。下面进行详细讲解。

15.1.1 边框合并属性 boder-collapse

边框合并属性(boder-collapse)用来定义表格中边框是独立显示还是合并显示。在边框合并属性中，使用的属性值有两个：separate 和 collapse。其语法结构如下所示。

```
border-collapse : separate | collapse;
```

其中部分属性值的含义如下所示。

- separate：定义表格边框独立显示。
- collapse：定义表格边框重叠显示。

下面是一个使用边框合并属性的实例，其代码如下所示。

例程 15-1 boder-collapse.html

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>边框合并属性</title>
<style>
table
{
border:3px solid #000000;
margin:0 0 18px 0;
}
.table1
{
border-collapse:separate;
}
.table2
{
border-collapse:collapse;
}
td
{
border:1px solid #999999;
width:200px;
height:80px;
}
</style>
```

```
11  </head>
12  <body>
    <table class="table1">
    <tr>
      <td>这里是表格中使用边框合并属性的情况</td>
      <td>这里是表格中使用边框合并属性的情况</td></tr>
      <tr>
      <td>这里是表格中使用边框合并属性的情况</td>
      <td>这里是表格中使用边框合并属性的情况</td></tr></table>
      <table class="table2">
      <tr>
      <td>这里是表格中使用边框合并属性的情况</td>
      <td>这里是表格中使用边框合并属性的情况</td></tr>
      <tr>
      <td>这里是表格中使用边框合并属性的情况</td>
      <td>这里是表格中使用边框合并属性的情况</td></tr></table>
13  </body>
14  </html>
```

以上的代码中，12 行中分别定义了两个两行两列的表格。07 行在第一个表格中，定义边框合并属性值为独立显示。08 行在第二个表格中，定义边框合并属性值为合并显示。同时 06 行定义了表格边框为 3px 黑色实线边框，09 行定义单元格为 1px 灰色实线边框。另外还定义了表格的边界属性，目的是将表格之间分隔一段距离。代码运行后的显示效果如图 15-1 所示。

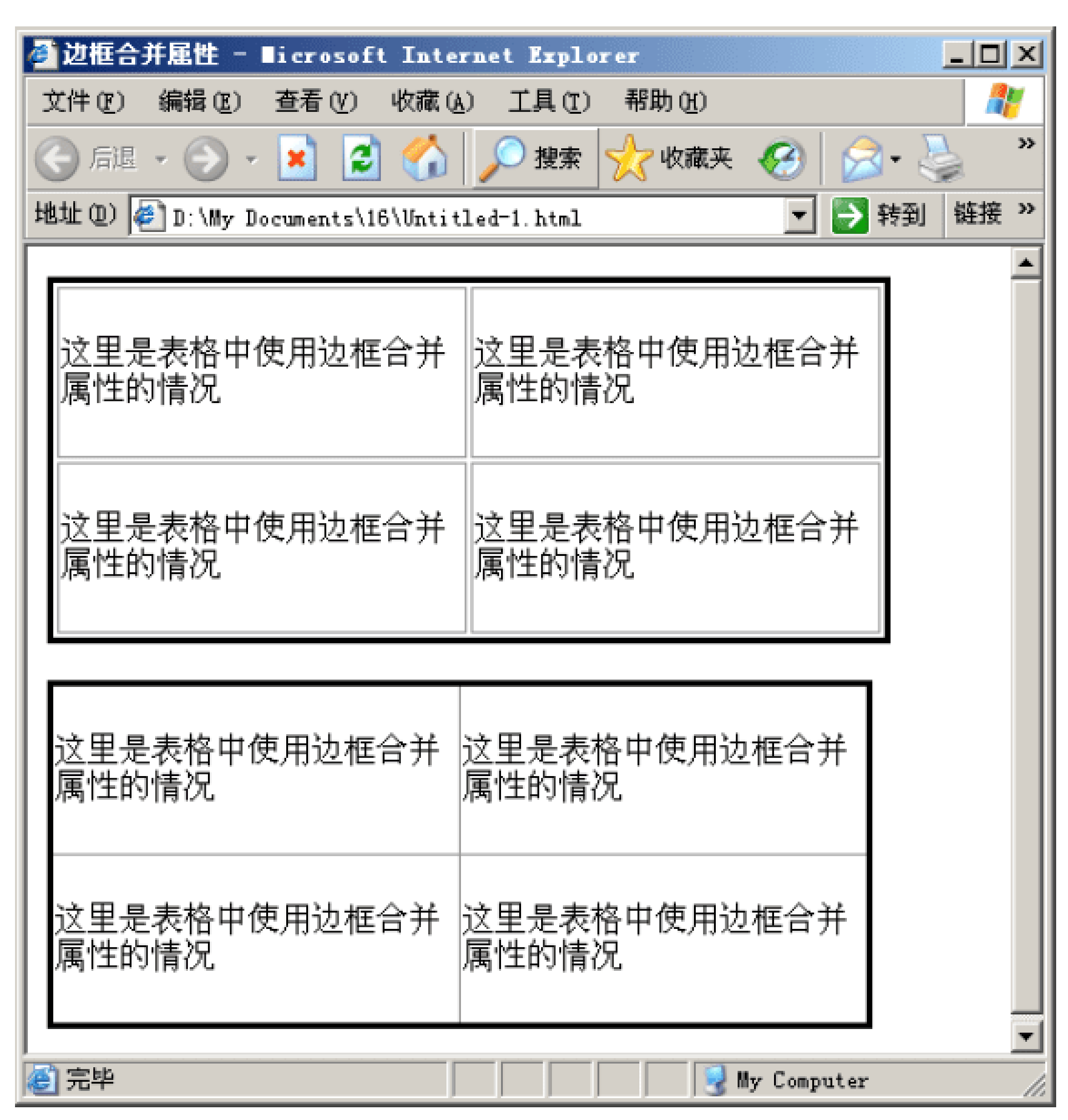


图 15-1　边框合并属性的显示效果

15.1.2　表格边框间距属性 border-spacing

表格边框间距属性(border-spacing)用来定义表格中独立边框之间的距离。在表格边框间距

属性中，使用的属性值是长度值。其语法结构如下所示。

```
border-spacing : length;
```



注意

当在表格边框合并属性中使用合并显示值(collapse)的时候，使用表格边框间距属性将不起作用。同时 IE 浏览器中并不支持表格边框间距属性，只有在 Firefox 等浏览器中，才能够正常显示出间距的效果。

下面是一个使用表格边框间距属性的实例，其代码如下所示。

例程 15-2　border-spacing.html

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>css 属性实例</title>
<style>
.table1
{
border-spacing:20px;
border-collapse:separate;
}
.table2
{
border-spacing:20px;
border-collapse:collapse;
}
table
{
border:2px solid #000000;
margin:0 0 20px 0;
}
td
{
border:1px solid #999999;
width:200px;
height:80px;
}
</style>
</head>
<body>
<table class="table1">
<tr><td>这里是表格中的内容部分</td><td>这里是表格中的内容部分</td></tr>
<tr><td>这里是表格中的内容部分</td><td>这里是表格中的内容部分</td></tr></table>
<table class="table2">
<tr><td>这里是表格中的内容部分</td><td>这里是表格中的内容部分</td>
</tr><td>这里是表格中的内容部分</td><td>这里是表格中的内容部分</td></tr>
```

```
13    </table>
14    </body>
15    </html>
```

以上的代码中，12 行中分别定义了两个两行两列的表格。06 行在第一个表格中，定义边框合并属性值为独立显示，并且定义了表格边框间距属性值为 20px。07 行在第二个表格中，定义了基本相同的属性，只是在表格边框合并属性中，定义属性值为合并显示。另外还定义了表格的边界属性，目的是将表格之间分隔一段距离。代码运行后，在 IE 浏览器中的显示效果如图 15-2 所示。在 Firefox 浏览器中的显示效果，如图 15-3 所示。

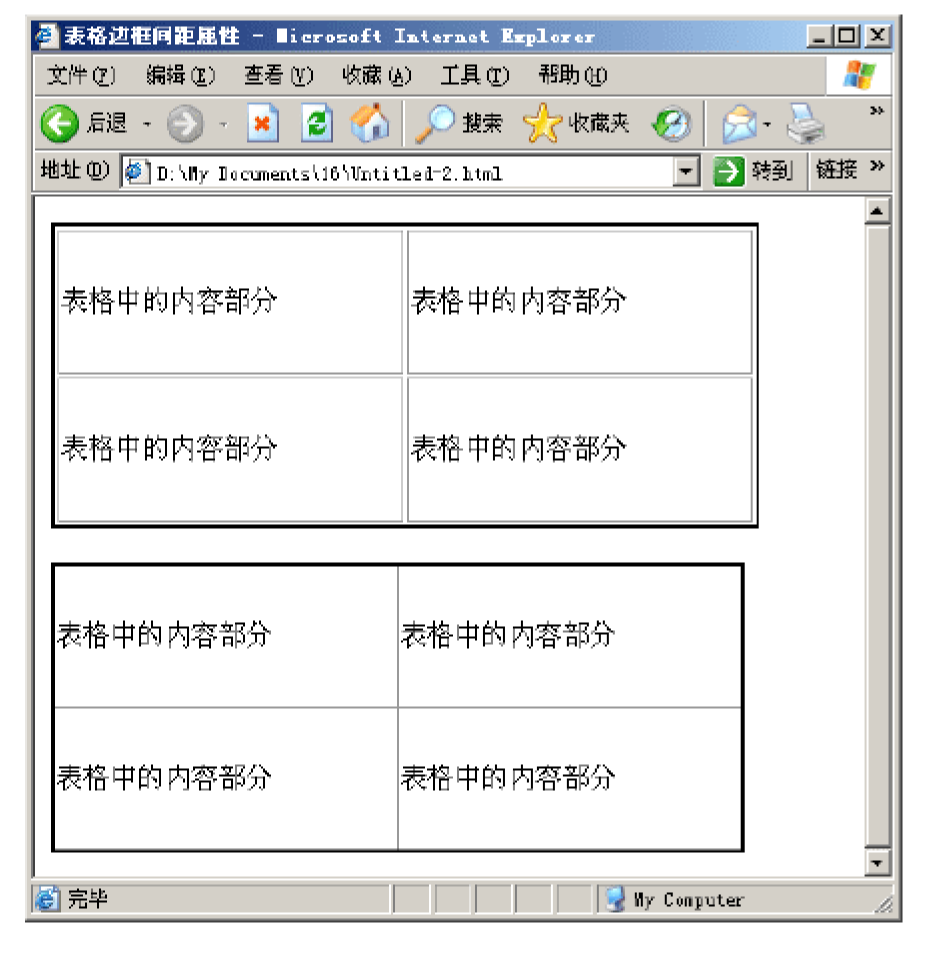


图 15-2　表格边框间距属性在 IE 中的效果

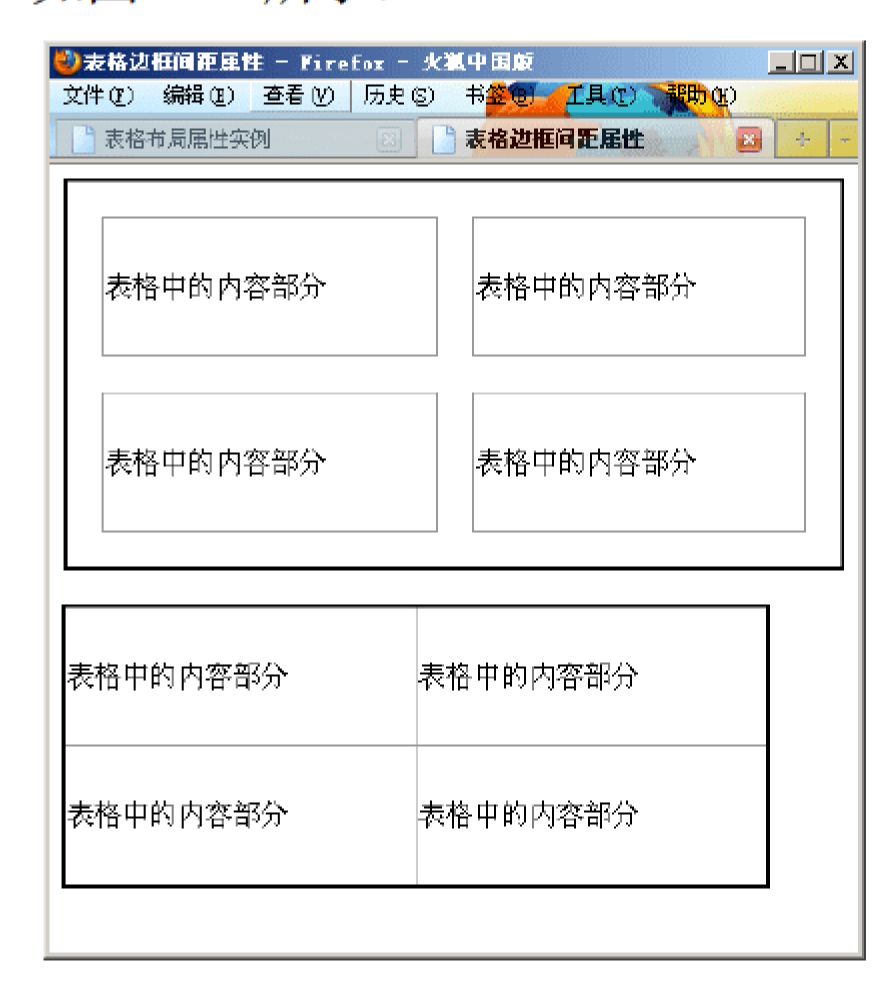


图 15-3　表格边框间距属性在 Firefox 中的效果

15.1.3　表格标题位置属性 caption-side

表格标题位置属性(caption-side)用来定义表格中标题元素<caption>显示的位置。在表格标题位置属性中，可以使用 4 个属性值：top、right、bottom 和 left。其语法结构如下所示。

```
caption-side : top | right | bottom | left;
```

注意

在 IE 浏览器中，不支持表格标题位置属性，只有在 Firefox 等浏览器中，才能够正常显示表格标题位置的变化。同时在表格中，必须首先定义标题元素<caption>，这样才能够正常更改标题元素的位置。

下面是一个使用表格标题位置属性的实例，其代码如下所示。

例程 15-3　caption-side.html

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
```

```
04  <title>css 属性实例</title>
05  <style>
06  .caption1
    {
    caption-side:top;
    }
07  .caption2
    {
    caption-side:right;
    }
    .caption3
08  {
    caption-side:bottom;
    }
09  .caption4
    {
    caption-side:left;
    }
10  table
    {
    border:2px solid #000000;
    margin:0 0 20px 0;
    }
11  td
    {
    border:1px solid #999999;
    width:200px;
    height:80px;
    }
12  </style>
13  </head>
14  <body>
    <table class="caption1">
    <caption>表格标题</caption>
    <tr><td>这里是表格中的内容部分</td><td>这里是表格中的内容部分</td></tr></table>
    <table class="caption2">
    <caption>表格标题</caption>
    <tr><td>这里是表格中的内容部分</td><td>这里是表格中的内容部分</td></tr></table>
    <table class="caption3">
    <caption>表格标题</caption>
    <tr><td>这里是表格中的内容部分</td><td>这里是表格中的内容部分</td></tr>
    </table>
    <table class="caption4">
    <caption>表格标题</caption>
    <tr><td>这里是表格中的内容部分</td><td>这里是表格中的内容部分</td></tr>
    </table>
15  </body>
16  </html>
```

以上的代码中，14 到 15 行中定义了 4 个两行的表格。06 到 10 行分别在 4 个表格中，定义了不同的表格标题位置属性值。代码运行后，在 IE 浏览器中的显示效果如图 15-4 所示。在 Firefox 浏览器中的显示效果如图 15-5 所示。

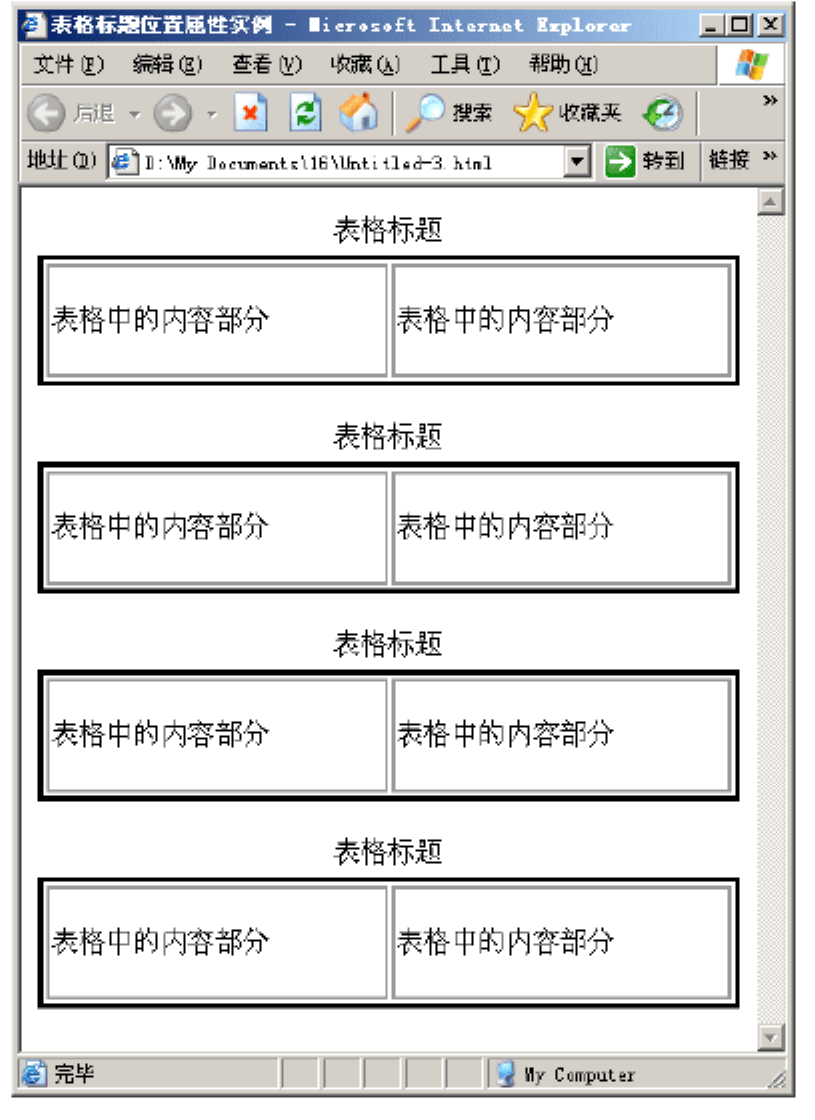



图 15-4　表格标题位置属性在 IE 中的效果

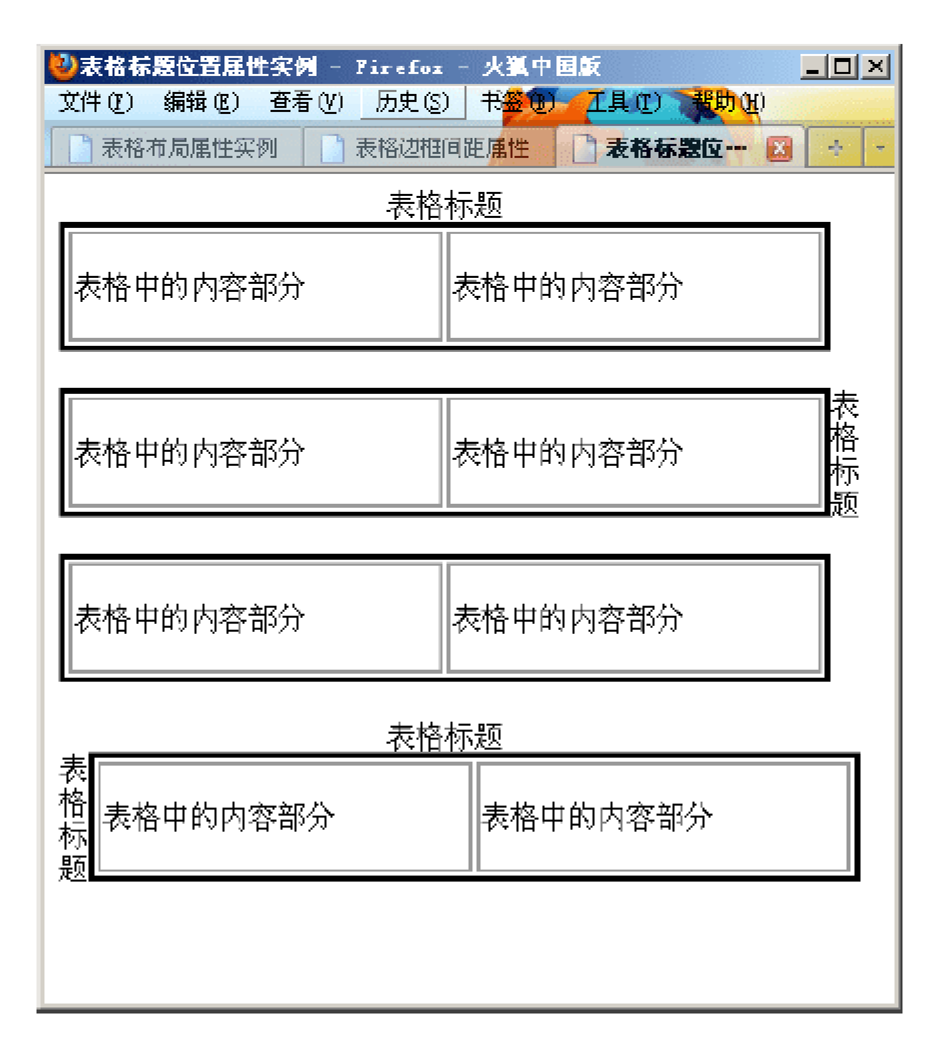


图 15-5　表格标题位置属性在 Firefox 中的效果

15.1.4　表格布局属性 table-layout

表格布局属性(table-layout)用来定义表格和表格内容之间的关系。在表格布局属性中，可以使用两个属性值：auto 和 fixed。其语法结构如下所示。

```
table-layout : auto | fixed;
```

其中各个属性值的含义，如下所示。

- auto：表格按照内容的多少自动分配各个单元格的宽度。
- fixed：表格按照单元格中定义的宽度显示表格内容。

下面是一个使用表格布局属性的实例，其代码如下所示。

例程 15-4　table-layout.html

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>css 属性实例</title>
<style>
.layout1
{
table-layout:auto;
}
.layout2
{
table-layout:fixed;
}
table
{
width:500px;
height:150px;
```

```
        border:2px solid #000000;
        margin:0 0 20px 0;
        }
09      td
        {
        border:1px solid #999999;
        width:50%;
        }
10      </style>
11      </head>
12      <body>
        <table class="layout1">
        <tr>
          <td><img src="images/pic.jpg" alt="pic" /></td>
          <td>assdssdsdabcdefghijklmnopqrs</td></tr>
        <tr>
          <td>这里是表格中的内容部分</td>
          <td>这里是表格中的内容部分</td></tr></table>
        <table class="layout2">
        <tr>
          <td><img src="images/pic.jpg" alt="pic" /></td>
          <td>assdssdsdabcdefghijklmnopqrs</td></tr>
        <tr>
          <td>这里是表格中的内容部分</td>
          <td>这里是表格中的内容部分</td></tr></table>
13      </body>
14      </html>
```

以上的代码中，06 和 07 行在两个<table>元素里定义了不同的表格布局属性值。同时统一定义了表格的宽度、高度、边框等属性，以及单元格边框和宽度属性。以上代码在 IE 浏览器中的显示效果如图 15-6 所示。在 Firefox 浏览器中的显示效果如图 15-7 所示。



图 15-6　表格布局属性在 IE 中的效果

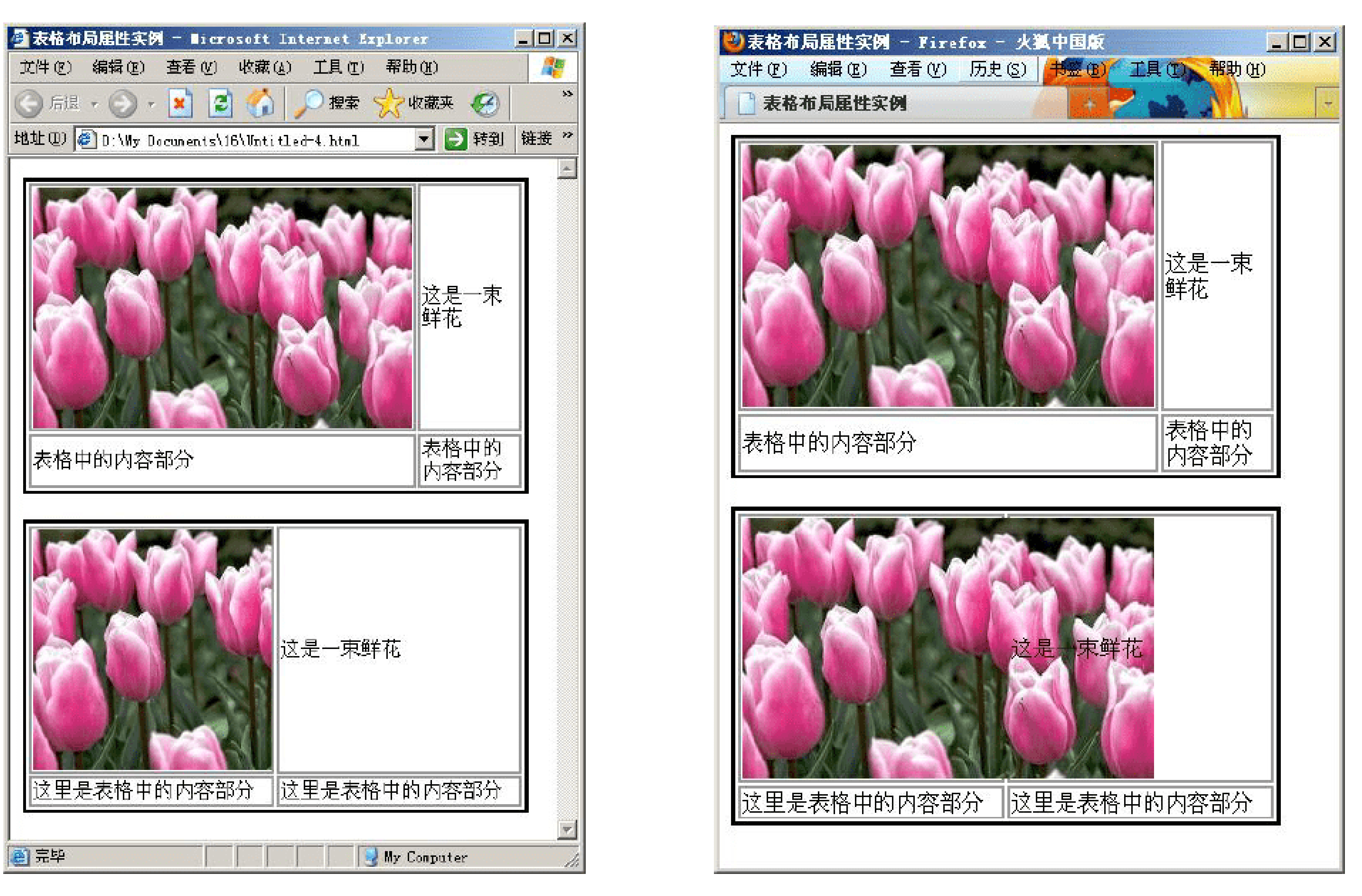


图 15-7　表格布局属性在 Firefox 中的效果

从图 15-6 可以看出，在 IE 浏览器中，如果定义表格布局属性值为 fixed，则单元格中的内容将会被裁剪，同时保持表格的总体宽度不变。从图 15-7 可以看出，在 Firefox 浏览器中，如果定义表格布局属性值为 fixed，则单元格中的内容不会被裁剪，同时两格单元格中的内容将发生重叠。每个单元格中的内容，按照各自原有的方式显示，并可以显示在表格之外。

15.2 单元格的制约关系

在使用表格元素的时候，通过定义 CSS 样式，可以控制单元格的高度、宽度等显示效果。当在某个单元格中定义了相应的属性，与该单元格相邻的单元格也将会受到影响。同时当多个单元格中，定义的高度和宽度与表格高度、宽度发生矛盾时，表格中单元格也会做相应的调整。其中具体的内容如下所示。

15.2.1 确定单行或列的宽度或高度

在使用表格元素的时候，由于表格中存在多个行或列，同时在每个单元格中都可以定义宽度和高度。当在某一列中，单元格中定义的宽度不同(并且不超过表格的宽度)时，会使用较大的宽度值。但是在某一行中，当定义某个单元格的高度属性值时，如果其他行没有定义任何高度属性，则按照最小的高度显示。

下面是一个同行或列中存在不同高度或宽度的实例，其代码如下所示。

例程 15-5 example.html

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>css 属性实例</title>
<style>
.td1
{
width:300px;
}
.td2
{
width:200px;
}
.td3
{
height:100px;
}
.td4
{
height:50px;
}
table
```

```
        {
        width:500px;
        height:100px;
        border:3px solid #000000;
        margin:0 0 20px 0;
        }
11      td
12      {
        border:2px solid #666666;
        }
13      </style>
14      </head>
15      <body>
        <table>
          <tr>
            <td class="td1">表格中的内容部分</td>
            <td>表格中的内容部分</td></tr>
        <tr>
            <td class="td2">表格中的内容部分</td>
            <td>表格中的内容部分</td></tr></table>
      <table>
        <tr>
            <td class="td3">表格中的内容部分</td>
            <td class="td4">表格中的内容部分</td></tr>
        <tr>
            <td>表格中的内容部分</td>
            <td>表格中的内容部分</td>
    </tr></table>
16      </body>
17      </html>
```

以上的代码中，06 和 07 行在第一个表格的第一列中，分别定义了两个宽度分别为 300px 和 200px 的单元格。08 和 09 行在第二个表格的第一行中，分别定义两个单元格的高度为 100px 和 50px。代码运行后的显示效果如图 15-8 所示。

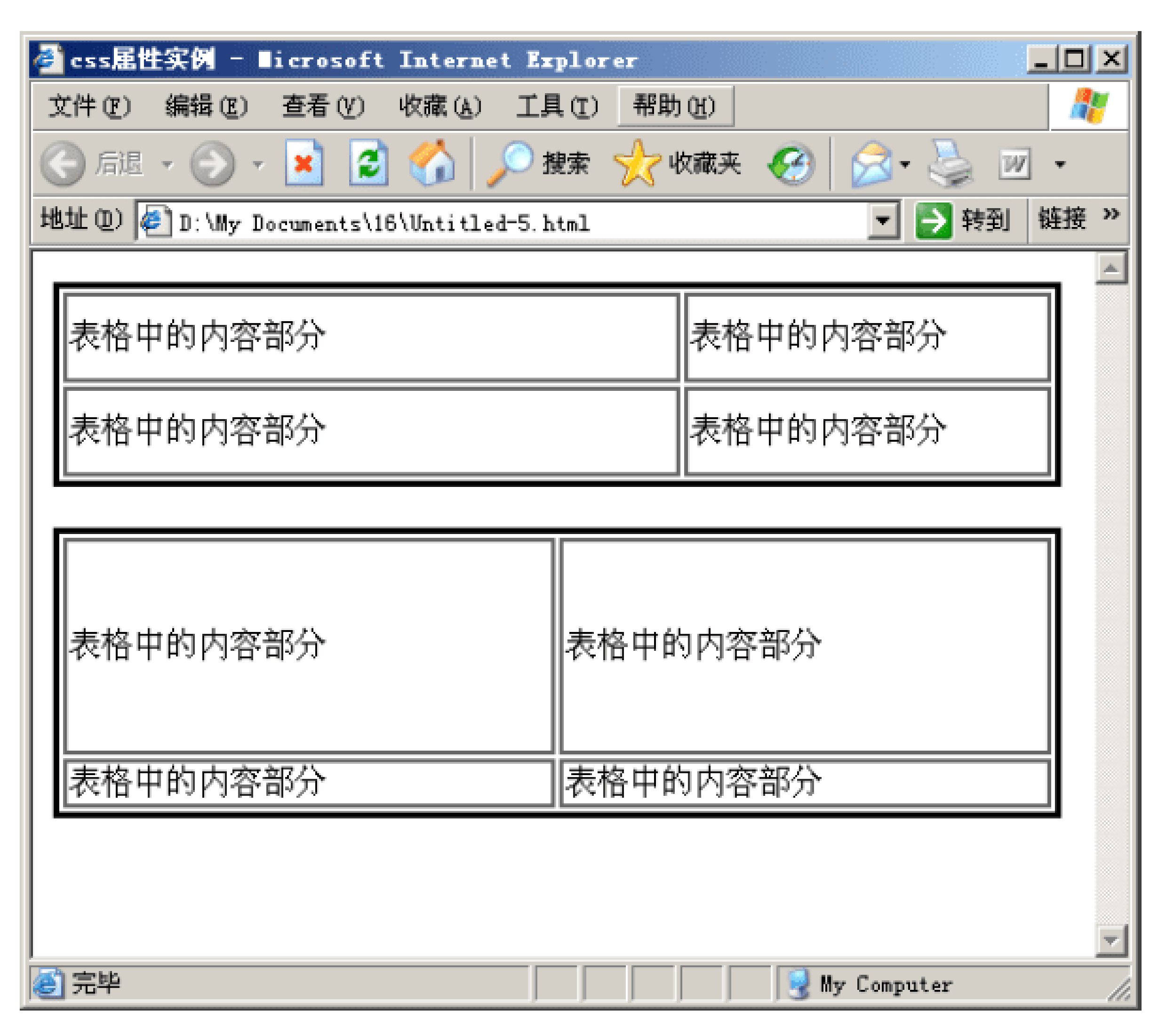


图 15-8　确定单行或列大小的显示效果

从图 15-8 可以看出，最终行或列的宽度或高度均和本行中定义的最大宽度或高度相同。同时在未合并任何单元格的情况下，表格同行或列均显示相同的宽度或高度。

15.2.2 确定多行或列的宽度或高度

如果在同个行或列中，每个单元格中都定义了宽度或者高度属性值，并且全部单元格的宽度或高度之和与表格的宽度或高度不相等，此时每个单元格将按照各自定义的宽度比例确定最终的宽度，而在高度方面，每个单元格均保持各自的高度。

下面是一个确定多行或列最终效果的实例，其代码如下所示。

例程 15-6 example.html

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>css 属性实例</title>
<style>
.td1
{
height:200px;
}
.td2
{
height:100px;
}
.td3
{
width:600px;
}
.td4
{
width:300px;
}
table
{
width:500px;
height:150px;
border:3px solid #000000;
margin:0 0 20px 0;
}
td
{
border:2px solid #999999;
}
</style>
</head>
<body>
<table>
```

```
    <tr>
      <td class="td1">科学部分</td>
      <td>人文部分</td></tr>
    <tr>
      <td class="td2">自然科学</td>
      <td>人物传记</td></tr></table>
  <table>
    <tr>
      <td class="td3">社会科学</td>
      <td class="td4">历史政治</td></tr>
    <tr>
      <td>哲学思想</td>
      <td>天文地理</td></tr></table>
15  </body>
16  </html>
```

以上的代码中，10 行定义两个表格的宽度均为 500px，高度均为 150px。06 和 07 行在第一个表格中，定义第一列中，两个单元格的高度为 200px 和 100px(此时高度之和大于表格高度)。08 和 09 行在第二个表格中，定义第一行中，两个单元格的宽度为 600px 和 300px(此时宽度之和大于表格宽度)。代码运行后的显示效果如图 15-9 所示。

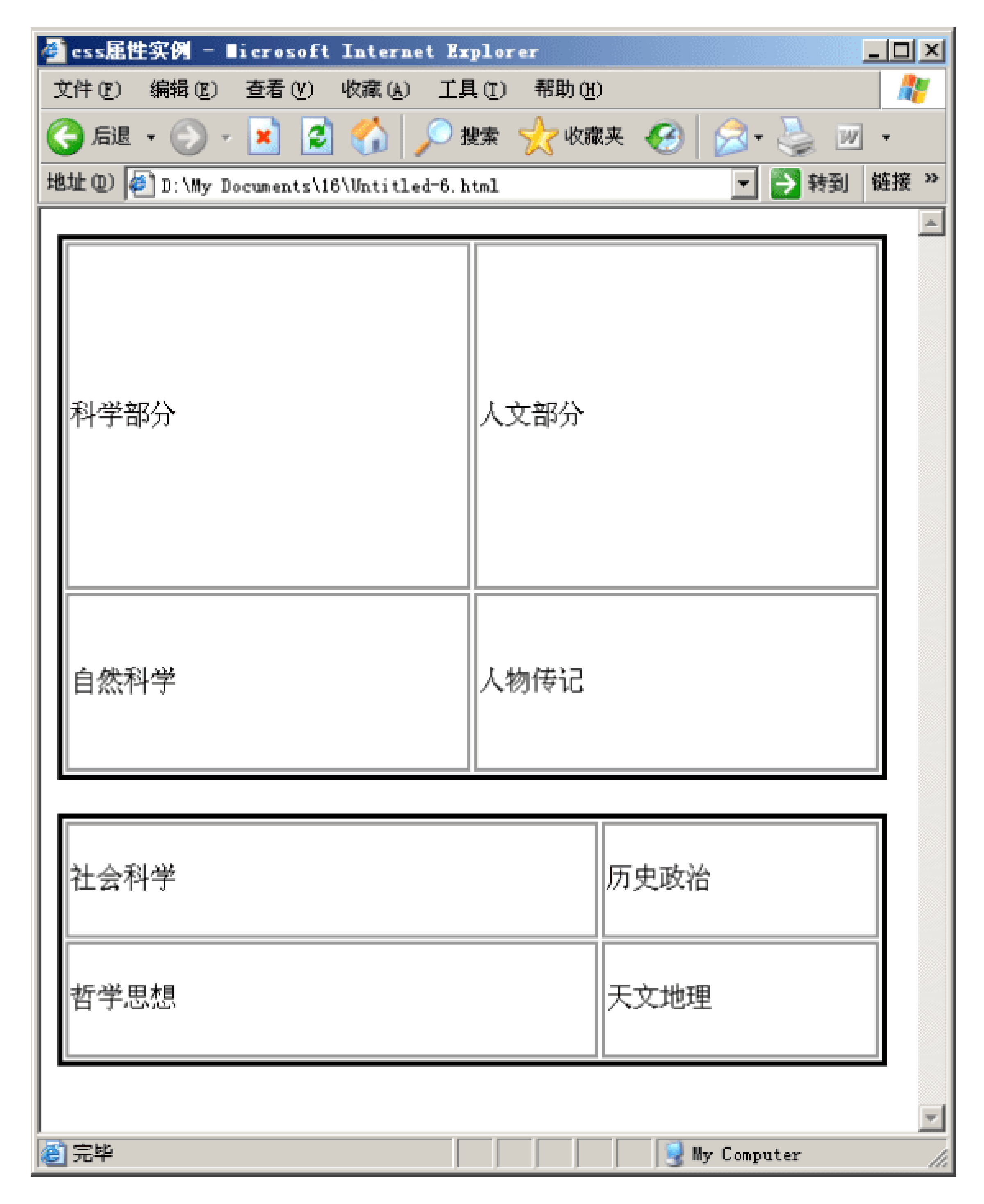


图 15-9　多行或列中定义大小的显示效果

从图 15-9 可以看出使用 td1 和 td2 时高度按比例分配，使用 td3 和 td4 的时候宽度按比例分配。

15.2.3 单元格与嵌套的<div>元素

在制作页面的时候，表格中单元格宽度和高度之间存在着制约的关系。当某个单元格内容变动的时候，相关联的行或列都会自动地适应单元格的变化。在使用嵌套的<div>元素进行布局的时候，并列的<div>元素之间，各自独立显示，无法适应其他<div>元素的变化。

下面是一个使用表格单元格和嵌套<div>元素的实例，其代码如下所示。

例程 15-7 example.html

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>css 属性实例</title>
<style>
.td1
{
background:#999999;
}
.td2
{
background:#FF0066;
}
.div1
{
float:left;
width:50%;
background:#999999;
}
.div2
{
float:left;
width:50%;
background:#FF0066;
}
.div0
{
width:500px;
border:2px solid #000000;
}
table
{
width:500px;
border:3px solid #000000;
margin:0 0 20px 0;
}
td
{
border:2px solid #999999;
}
</style>
</head>
<body>
```

```
<table>
  <tr>
    <td class="td1">这里是表格中的内容部分</td>
    <td class="td2">这里是表格中的内容部分<br />这是添加的部分<br />这是添加的部分<br />这是添加的部分</td></tr>
  </table>
  <div class="div0">
    <div class="div1">这里是表格中的内容部分</div>
    <div class="div2">这里是表格中的内容部分<br />这是添加的部分<br />这是添加的部分<br />这是添加的部分</div>
  </div>
</body>
</html>
```

以上的代码中，在<table>元素和嵌套<div>元素的父元素中，定义了相同的宽度、边框属性值。在单元格和嵌套的子元素中，定义了相同的背景颜色。当右侧单元格或者子元素中添加内容后，页面的显示效果如图 15-10 所示。

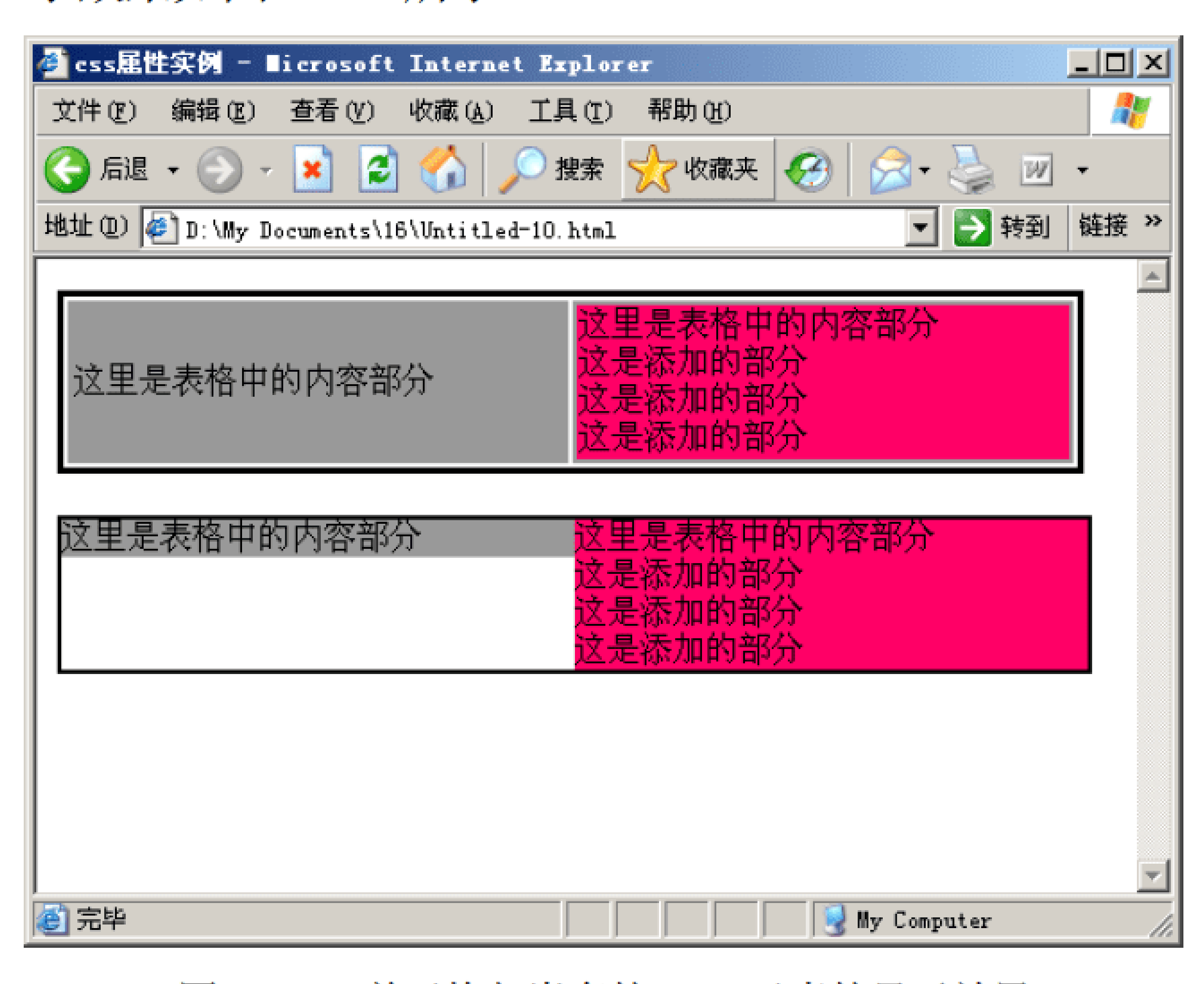


图 15-10　单元格与嵌套的<div>元素的显示效果

从图 15-10 可以看出，在表格中，由于单元格的制约关系，当一侧的单元格中内容增加时，另一侧的背景会随之增加。但在<div>元素构成的结构中，由于各个<div>相互独立，所以背景也各自独立显示。

15.3 本章习题

一、选择题

1．以下关于表格边框间距属性(border-spacing)关系说法中，错误的是：(　　)

A. IE 浏览器中，并不支持表格边框间距属性。

B. 在 Firefox 浏览器中，能够正常显示出间距的效果。

C. 表格边框间距属性使用的属性值是长度值。

D. 表格边框间距属性的效果在 IE 浏览器中也可以显示。

2．以下有关表格标题位置属性(caption-side)定义中，正确的是：(　　)

A. IE 浏览器中，表格标题位置属性作用效果会正确显示。

B. 在 Firefox 浏览器中，表格标题位置属性作用效果会正确显示。

C. 在表格中，不定义标题元素<caption>，也能够正常更改标题元素的位置。

D. caption{caption-side:bottom;}在 IE 浏览器中，标题会显示在表格的下方。

二、填空题

1．边框合并属性(boder-collapse)用来定义______________________。在边框合并属性中，使用的属性值有_________和__________。

2．表格标题位置属性(caption-side)用来定义_________________。在表格标题位置属性中，可以使用 4 个属性值：_____、______、_______、_______。

3．表格布局属性(table-layout)中，可以使用两个属性值：________和_______，分别表示_________________和___________________________。

三、实战练习

1．制作两个两行两列的表格。在第一个表格中，定义边框合并属性值为独立显示，并且定义了表格边框间距属性值为 50px。在第二个表格中，表格边框合并属性值为合并显示。分别在 IE 浏览器和 Firefox 浏览器中显示，注意它们的效果。

2．制作一个表格，要求表格标题在表格的右边显示。分别在 IE 浏览器和 Firefox 浏览器中显示，注意它们的效果。

3．使用表格布局属性 table-layout 制作两行两列表格，要求在其中的一个单元格中使用图片。分别在 IE 浏览器和 Firefox 浏览器中显示，注意它们的效果。

Chapter 16

CSS控制元素的大小

在网页中，一个元素占有空间的大小，由几个部分构成。其中包括元素的内容、元素的补白、元素的边框、元素的边界等 4 个部分。在这 4 个部分占有的空间中，有的部分可以显示相应的内容，而有的部分只用来分隔相邻的元素或区域。4 个部分一起构成了 CSS 中元素的盒模型。合理控制元素实际占有空间的大小，有助于更好地对各个元素进行合理的布局。

本章主要内容有:

◎ 理解盒模型的概念。
◎ 能够熟练控制元素内容的大小。
◎ 熟悉页面背景的各个属性。
◎ 重点掌握元素的补白和边界的特点及其使用方法。
◎ 熟悉元素的背景和边框的属性及其应用方法。

16.1 盒模型的概念

一个块元素可以看成是一个盒子形状的模型。这个模型的主体部分是元素的内容部分，用来显示元素中的主要信息，这个部分由 width(宽度)属性和 height(高度)属性来控制。在内容部分之外是补白部分，由 padding(补白)属性控制。在补白部分之外是元素的边框，由 border(补白)属性控制。在盒模型的最外面是边界部分，由 margin(补白)属性控制。盒模型的具体显示效果，如图 16-1 所示。

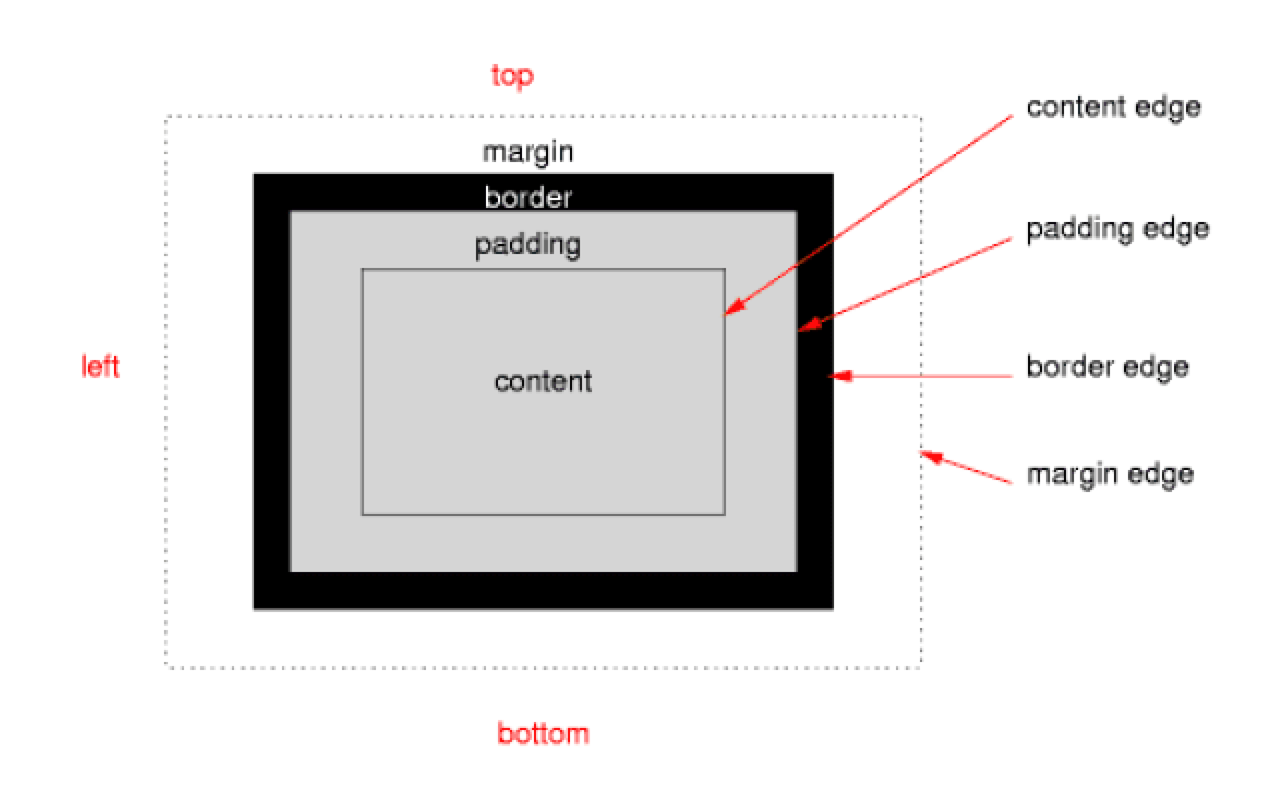


图 16-1　盒模型的示意图

可以发现，一个元素所占有的空间，与盒模型中 4 个部分各自占有空间有关。由于在 CSS 中，可以分别定义每个方向上的盒模型属性(内容部分除外)。所以整个元素占有空间的宽度，使用下面的公式计算。

```
左边界+左边框+左补白+宽度+右补白+右边框+右边界
```

整个元素占有空间的高度，使用下面的公式计算。

```
上边界+上边框+上补白+高度+下补白+下边框+下边界
```

在使用 CSS 布局时，要随时注意计算元素占有的空间。在盒模型中，除 width(宽度)属性和 height(高度)属性包含的内容部分之外，其他的区域中，可能不显示任何内容(可能会显示元素自身，或者其他元素的背景)。这就使元素占有区域大小含有很大的隐蔽性，从而给合理布局元素带来很大的困难。

16.2 元素内容的大小

元素内容的大小，用来控制元素内容的显示效果。通常使用宽度属性(width)和高度属性

(height)来定义。有时候(例如，对元素的适应性有特殊要求的时候)还要使用其他的几个尺寸属性，包括 max-width、min-width、max-height、min-height 等。下面分别进行讲解。

16.2.1　宽度属性 width

宽度属性(width)用来定义元素内容的宽度。在宽度属性中，可以使用 3 种属性值，分别为 auto 值、长度值和百分比值。其语法结构如下所示。

```
width:auto | length | percent;
```

下面是一个使用宽度属性的实例，其代码如下所示。

例程 16-1　width.html

```
01  <html>
02  <head>
03  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
04  <title>无标题文档</title>
05  <style>
06  div{
    width:300px;
    background:#CCFF99;}
07  </style>
08  </head>
09  <body>
    <div>这是一个 width 属性的实例</div>
10  </body>
11  </html>
```

以上的代码中，06 行在 div 选择符中，使用宽度属性定义元素的宽度为 300px，同时还定义了 background 属性来定义元素的背景颜色，以便能够更好地显示元素的大小。以上代码的显示效果，如图 16-2 所示。

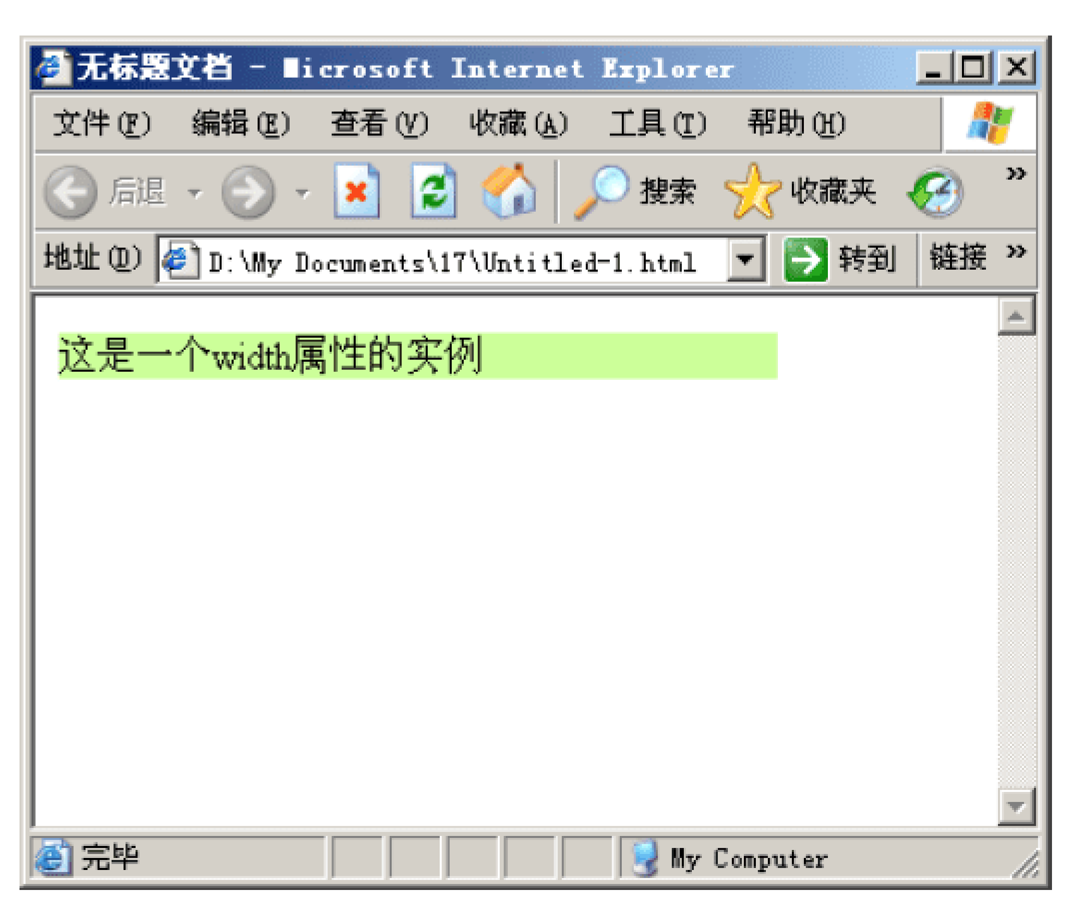


图 16-2　宽度属性的显示效果

16.2.2　高度属性 height

高度属性(height)用来定义元素内容的高度。在高度属性中，可以使用 3 种属性值，分别为 auto 值、长度值和百分比值。其语法结构如下所示。

```
height:auto | length | percent;
```

下面是一个使用 height 属性的实例，其代码如下所示。

例程 16-2　height.html

```
01   <html >
02   <head>
03   <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
04   <title>无标题文档</title>
05   <style>
06   div{
     height:120px;
     width:300px;
     background:#FFFF33;}
07   </style>
08   </head>
09   <body>
     <div>这是一个高度属性的实例</div>
10   </body>
11   </html>
```

以上的代码中，06 行使用高度属性定义元素内容的高度为 120px，同时使用宽度属性定义元素内容的宽度为 300px。此时元素中的内容并没有超出元素定义的大小时，元素内容部分会保持定义的大小，在没有内容的空白区域将显示元素的背景(当元素有嵌套关系时，也可能显示父元素的背景)。以上代码的显示效果，如图 16-3 所示。

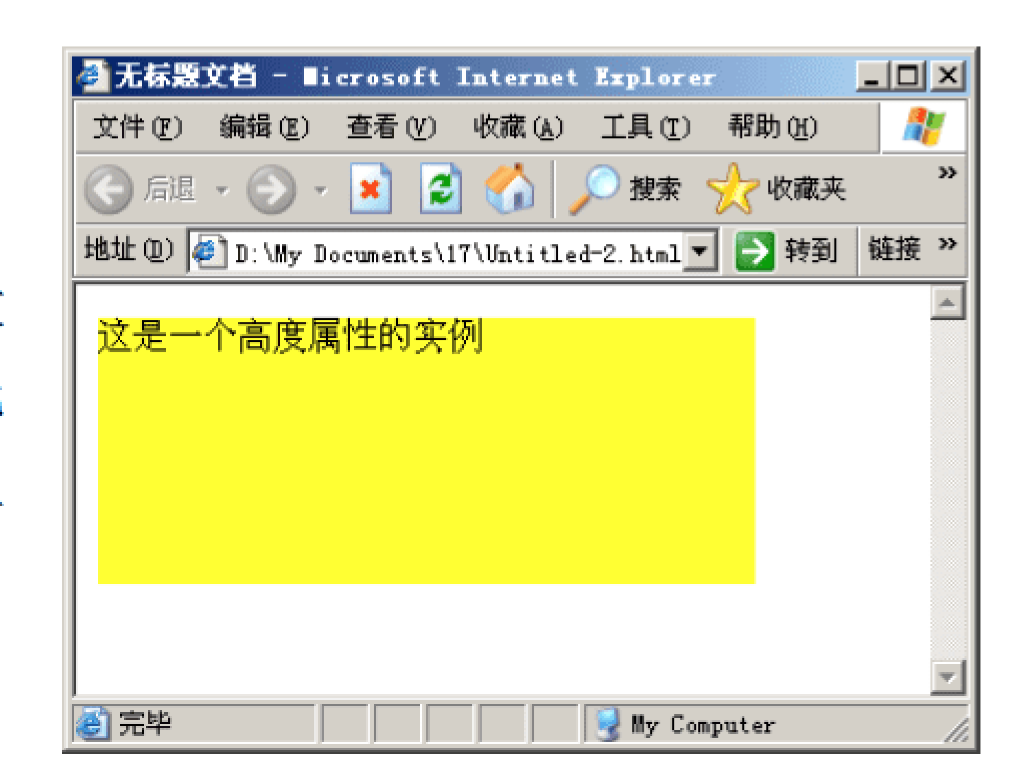


图 16-3　高度属性的显示效果

16.2.3　内容与宽度、高度属性的关系

内容与宽度、高度属性的关系，和使用的浏览器版本有关。这里以 IE 6.0 为例来讲解两者之间的关系。

在 IE 6.0 中，元素中的文本内容在默认的情况下(即未定义任何显示属性的情况下)，元素中的文本内容，会按照元素定义的宽度属性自动换行显示。当元素中的文本内容超出元素定义的高度属性值时，元素会自动更改高度，以适应内容的增加。

下面是一个元素内容超出元素定义大小的实例，其代码如下所示。

例程 16-3　example.html

```
01   <html >
02   <head>
```

```
03  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
04  <style>
05  div{
    height:60px;
    width:200px;
    background:#666666;}
06  p{
    position:absolute;
    left:320px;
    top:15px;
    height:60px;
    width:200px;
    background:#FFFF33;}
07  </style>
08  </head>
09  <body>
    <div>元素内容超出元素定义大小的实例，注意观察元素背景的延伸情况，这将有助于理解元素内容
与元素大小之间的关系。</div>
    <p>这里是对比内容</p>
10  </body>
11  </html>
```

以上的代码中，05 行中定义了元素内容的大小为 200px×60px，背景颜色为灰色。同时 06 行定义了一个相同大小的<p>元素作为参照。以上代码的显示效果，如图 16-4 所示。

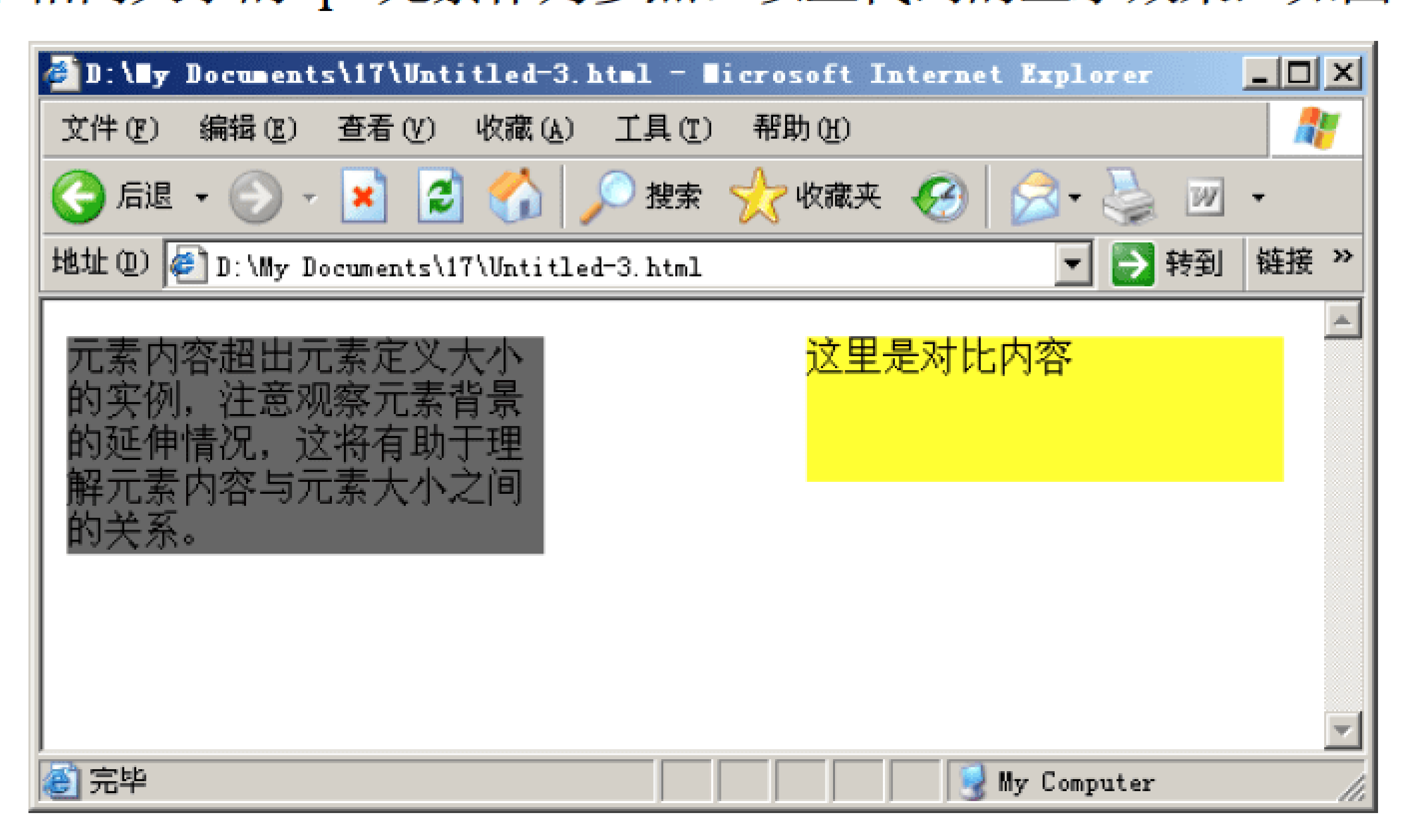


图 16-4　内容超出元素定义大小的显示效果

16.3 元素的背景

元素的背景通常用来显示修饰的内容，可以在背景中定义背景颜色，也可以在背景中定义图像内容。在 CSS 中，背景属性包括：背景颜色属性(background-color)、背景图片属性(background-image)、背景图片重复属性(background-repeat)、背景图片位置属性(background-position)、背景图片滚动属性(background-attachment)、背景综合属性(background)

等。下面分别进行讲解。

16.3.1　背景颜色属性 background-color

背景颜色属性(background-color)用来定义元素的背景颜色。在背景颜色属性中，可以使用两种属性值，分别为颜色值和颜色名称。其语法结构如下所示。

```
background-color:color | colorname;
```

下面是一个使用背景颜色属性的实例，其代码如下所示。

例程 16-4　background-color.html

```
01  <html>
02  <head>
03  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
04  <title>css 属性实例</title>
05  <style>
06  div{
    background:gray;
    height:100px;
    width:400px;}
07  </style>
08  </head>
09  <body>
    <div>这是灰色背景颜色的实例。</div>
10  </body>
11  </html>
```

在以上的代码中，06 行定义元素的背景颜色属性值为灰色，同时定义了元素的宽度和高度，方便背景颜色的显示。以上代码的显示效果如图 16-5 所示。

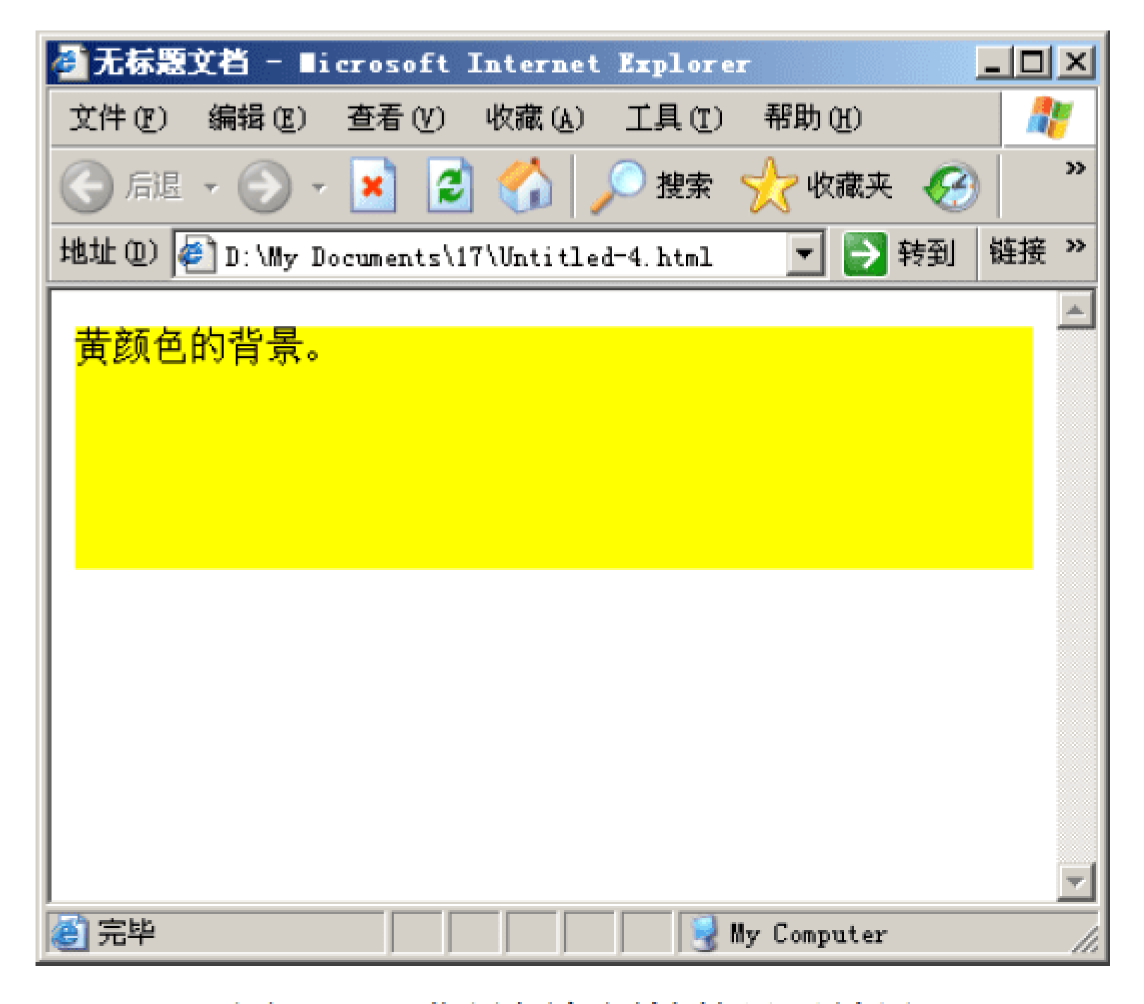


图 16-5　背景颜色属性的显示效果

16.3.2 背景图片属性 background-image

背景图片属性(background-image)用来定义元素的背景图片。在背景图片属性中，要使用图片文件所在的 url 作为属性的值。其语法结构如下所示。

```
background-image:url;
```

下面是一个使用背景图片属性的实例，其代码如下所示。

例程 16-5 background-image.html

```
01  <html >
02  <head>
03  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
04  <title>无标题文档</title>
05  <style>
06  div{
    background-image:url(images/pic.jpg);
    height:100px;
    width:400px;
  }
07  </style>
08  </head>
09  <body>
    <div>这是背景图。</div>
10  </body>
11  </html>
```

在以上的代码中，06 行定义了背景图片属性，同时定义了元素的宽度和高度，方便背景图片的显示。在没有定义背景图片的任何显示属性的时候，背景图片会以元素的左上角为中心，重复排列显示。以上代码的显示效果如图 16-6 所示。

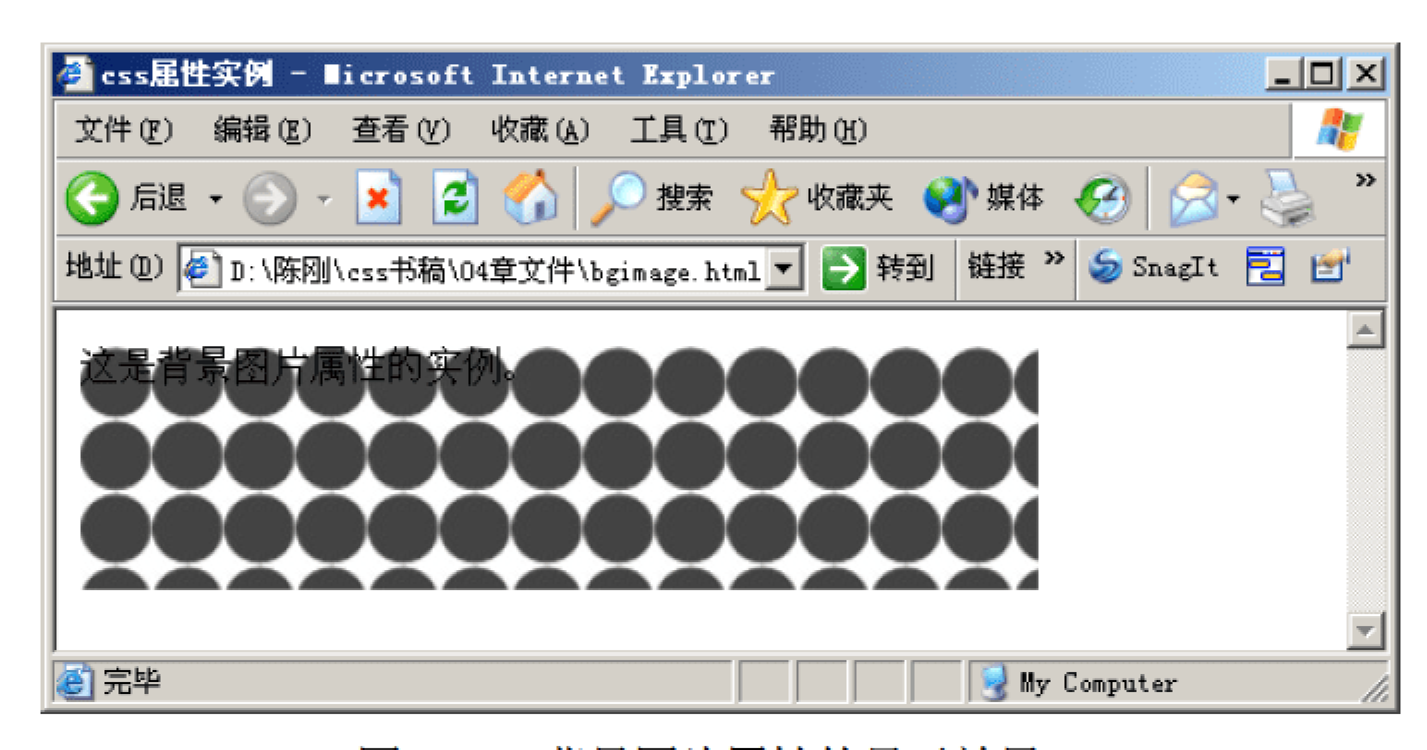


图 16-6 背景图片属性的显示效果

16.3.3 背景图片的重复属性 background-repeat

背景图片的重复属性(background-repeat)用来定义元素背景图片的重复排列方式。在背景图

片的重复属性中，可以使用 4 个属性值，分别为 repeat、no-repeat、repeat-x 和 repeat-y。其语法结构如下所示。

```
background-repeat: repeat | no-repeat | repeat-x | repeat-y;
```

4 个属性值的含义如下所示。

- repeat：背景图片重复排列。
- no-repeat：背景图片不重复排列。
- repeat-x：背景图片沿 X 轴重复排列。
- repeat-y：背景图片沿 Y 轴重复排列。



注意

在使用背景图片的重复属性的时候，一定要同时使用背景图片属性。否则定义的背景图片的重复属性将不起作用。

下面是一个使用背景图片重复属性的实例，其代码如下所示。

例程 16-6 background-repeat.html

```
01  <html >
02  <head>
03  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
04  <title>无标题文档</title>
05  <style>
06  div{
    background-repeat:repeat-x;
    background-image:url(images/pic.jpg);
    height:100px;
    width:400px;}
07  </style>
08  </head>
09  <body>
    <div>这是背景图片重复属性的实例。</div>
10  </body>
11  </html>
```

在以上的代码中，06 行定义了背景图片，同时定义了图片的重复属性为纵向重复。在设置纵向重复后，背景图片会以元素左侧为基准，纵向重复排列显示。以上代码的显示效果如图 16-7 所示。

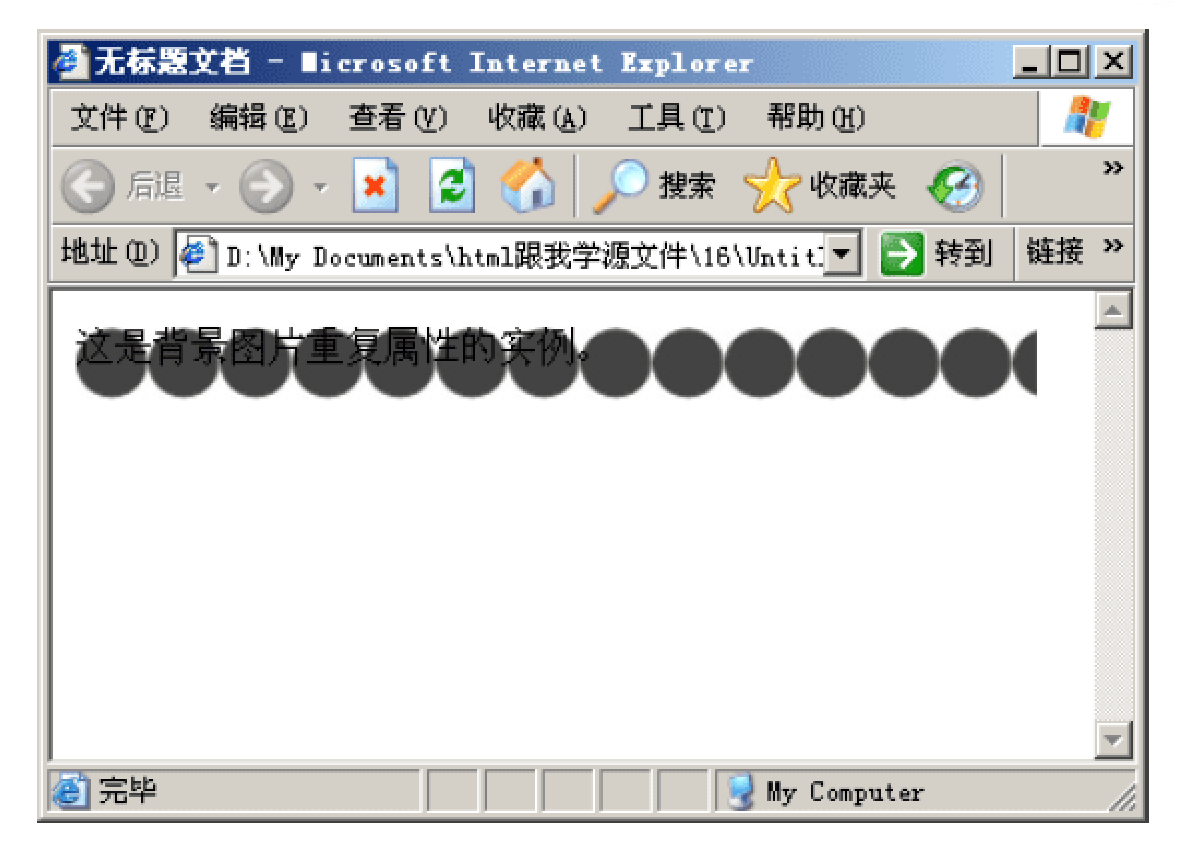


图 16-7 背景图片重复属性的显示效果

16.3.4 背景图片位置属性 background-position

背景图片位置属性(background-position)用来定义元素背景图片的起始位置。在背景图片位置属性中，可以使用两种属性值：一种为长度单位，包括长度值和百分比值；另一种为制定值，包括 top、center、bottom、left、center、right 等。其语法结构如下所示。

```
background-repeat: length | percent | top | center | bottom | left | right;
```

其中部分属性值的含义如下所示。

- top：背景图片的初始位置为元素顶部。
- center：背景图片的初始位置为元素中部。
- left：背景图片的初始位置为元素左侧。
- right：背景图片的初始位置为元素右侧。
- bottom：背景图片的初始位置为元素底部。

在背景图片位置属性中，属性值一般有两个，前一个代表横向位置，后一个代表纵向位置。使用两个属性值，定义背景图片的起始位置。如果两个方向的起始位置相同，则可以使用一个值。

下面是一个使用背景图片位置属性的实例，其代码如下所示。

例程 16-7 background-position.html

```
<html >
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>无标题文档</title>
<style>
div{
background-position:center right;
background-repeat:repeat-y;
background-image:url(images/pic.jpg);
height:100px;
```

```
        width:400px;}
07   </style>
08   </head>
09   <body>
     <div>这是背景图片位置属性的实例。</div>
10   </body>
11   </html>
```

在以上的代码中，06 行定义了背景图片的位置为中间、右部。同时定义了背景图片为纵向重复。此时背景图片会按照元素的右边为起点，按照背景图片重复属性中定义的属性值重复排列。以上代码的显示效果如图 16-8 所示。

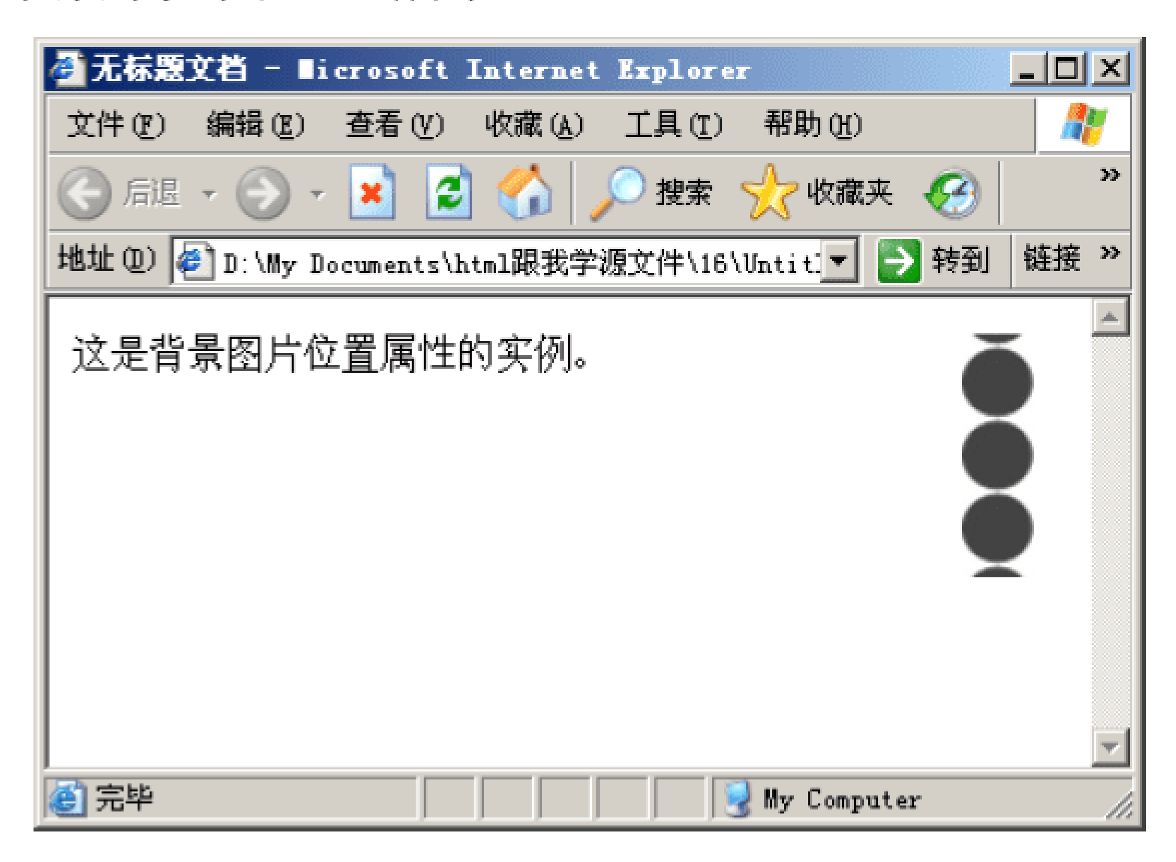


图 16-8　背景图片位置属性的显示效果

16.3.5　背景图片滚动属性 background-attachment

背景图片滚动属性(background-attachment)用来定义元素背景图片是否随浏览器滚条的拖动而滚动。在背景图片滚动属性中，可以使用两个属性值，分别为 scroll 和 fixed。其语法结构如下所示。

```
background- attachment: scroll | fixed;
```

其中各个属性值的含义如下所示。

- scroll：背景图片随滚条的变化而滚动。
- fixed：背景图片固定不动。

下面是一个使用背景图片滚动属性的实例，其代码如下所示。

例程 16-8　background-attachment.html

```
<html >
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>无标题文档</title>
<style>
body{
```

```
        background-attachment:scroll;
        background-position:center;
        background-repeat:no-repeat;
        background-image:url(images/pic.jpg);
    }
07  div{
    height:400px;}
08  </style>
09  </head>
10  <body>
     <div>背景图片滚动属性的实例。</div>
11  </body>
12  </html>
```

在以上的代码中，06 行定义了背景图片的滚动属性值为滚动。同时定义了背景图片为不重复，居中显示。当拖动浏览器右侧的滚条时，背景图片的位置将发生改变。以上代码的在浏览器中运行后，当拖动滚条时，显示效果分别如图 16-9 和图 16-10 所示。

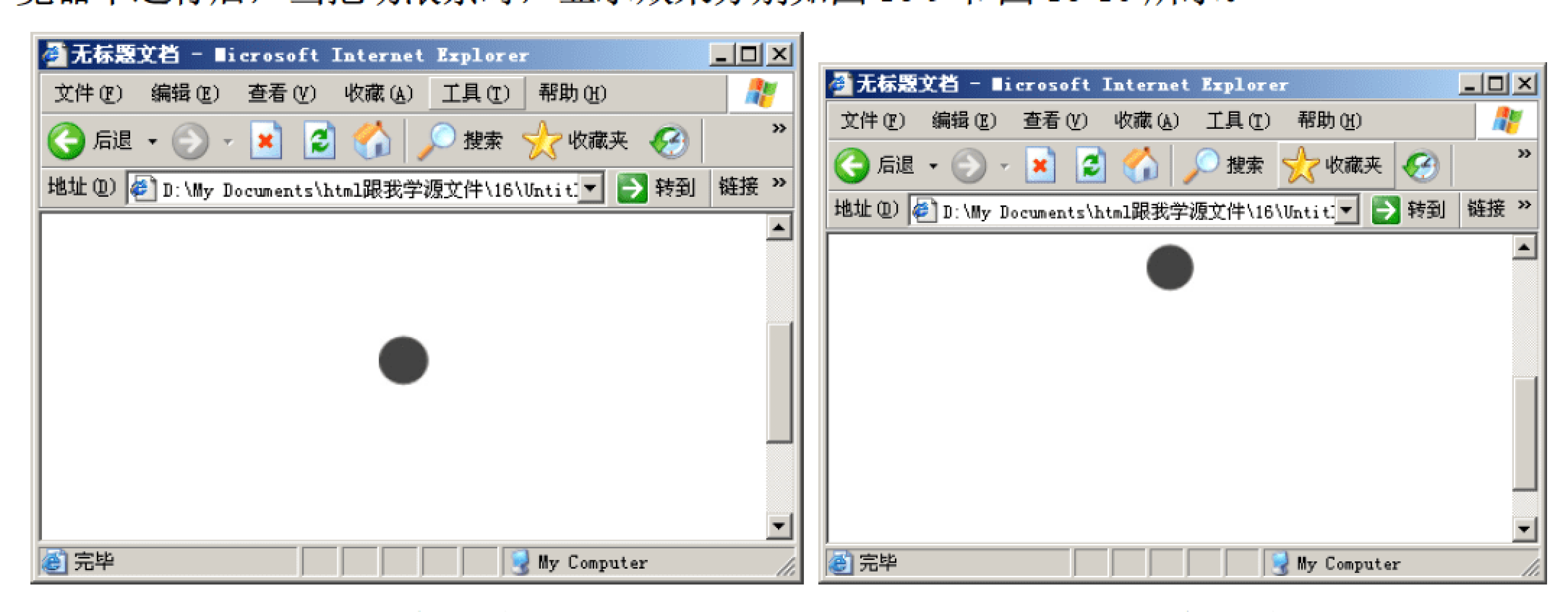


图 16-9　背景图片滚动属性的显示效果 1　　图 16-10　背景图片滚动属性的显示效果 2

16.3.6　同时使用背景颜色和背景图片属性

当同时使用背景颜色和背景图片属性时，背景图片会覆盖背景颜色。如果在背景图片属性中还使用了其他的相关属性，则在背景图片显示范围之外显示背景颜色。下面是一个同时使用背景颜色和背景图片属性的实例，其代码如下所示。

例程 16-9　example.html

```
01  <html >
02  <head>
03  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
04  <title>无标题文档</title>
05  <style>
06  div{
    background-color:yellow;
    background-position:center;
```

```
        background-repeat:repeat-y;
        background-image:url(images/pic.jpg);
        width:300px;
        height:100px;
        }
07  </style>
08  </head>
09  <body>
    <div>同时使用背景颜色和背景图片属性的实例。</div>
10  </body>
11  </html>
```

在以上的代码中，06 行定义了背景颜色为黄色。同时定义了背景图片为居中，纵向重复显示。此时纵向排列的背景图片将覆盖黄色的背景内容。由于图片在横向不重复，所以在背景图片的两侧会显示背景颜色。以上代码的显示效果如图 16-11 所示。

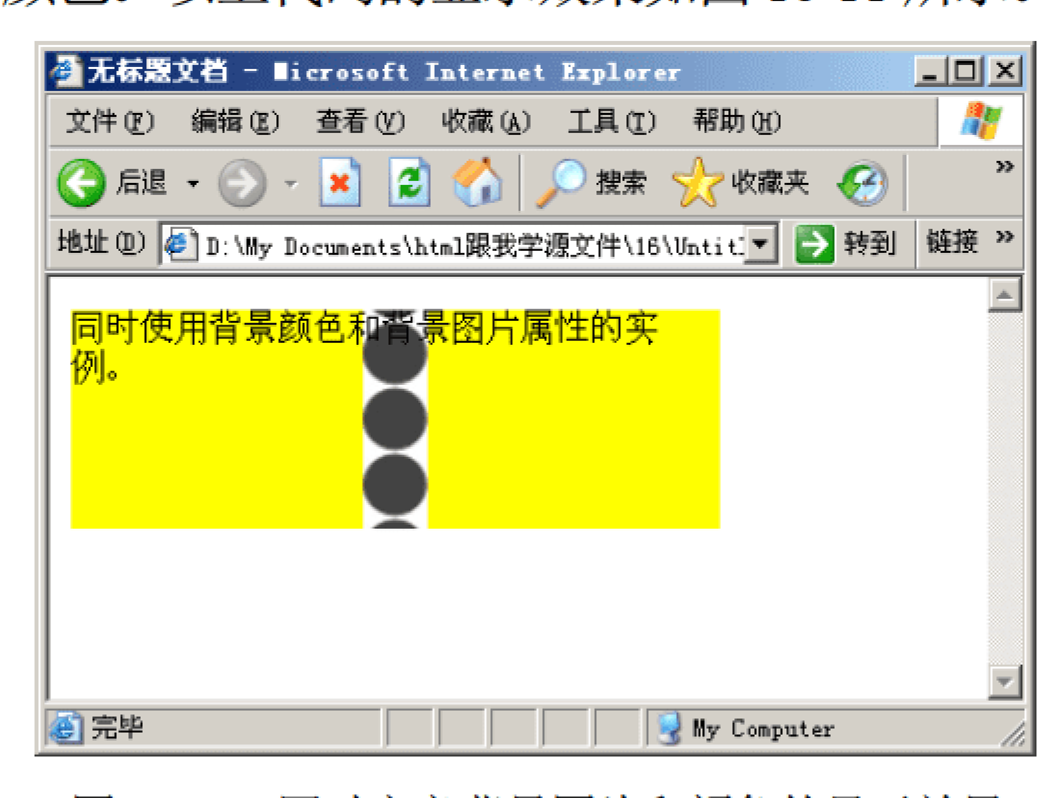


图 16-11　同时定义背景图片和颜色的显示效果

16.3.7　背景的综合属性 background

背景的综合属性(background)用来同时定义元素背景的各种显示属性。在实际应用 CSS 的过程中，通常使用背景的综合属性一次定义所有的背景属性。

背景的综合属性的语法如下所示。

```
    Background: background-image background-repeat background-position background-color
background-attachment:
```

在背景的综合属性中，所有的背景属性值之间，使用空格分隔。当属性值的位置发生变化时，并不影响背景属性的显示效果。

下面是一个同时使用背景的综合属性的实例，其代码如下所示。

例程 16-10　background.html

```
<html >
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>css 属性实例</title>
```

```
05    <style>
06    div{
      background:url(images/round.jpg) repeat-x center #999999;
      width:400px;
      height:100px;
      }
07    </style>
08    </head>
09    <body>
      <div>这是背景的综合属性的实例。</div>
10    </body>
11    </html>
```

在以上的代码中，06 行使用了背景综合属性，一次性定义了：背景图片的路径为“images/round.jpg”，背景图片的重复属性值为横向重复，背景图片的位置为居中，背景颜色为黄色。以上代码的显示效果如图 16-12 所示。

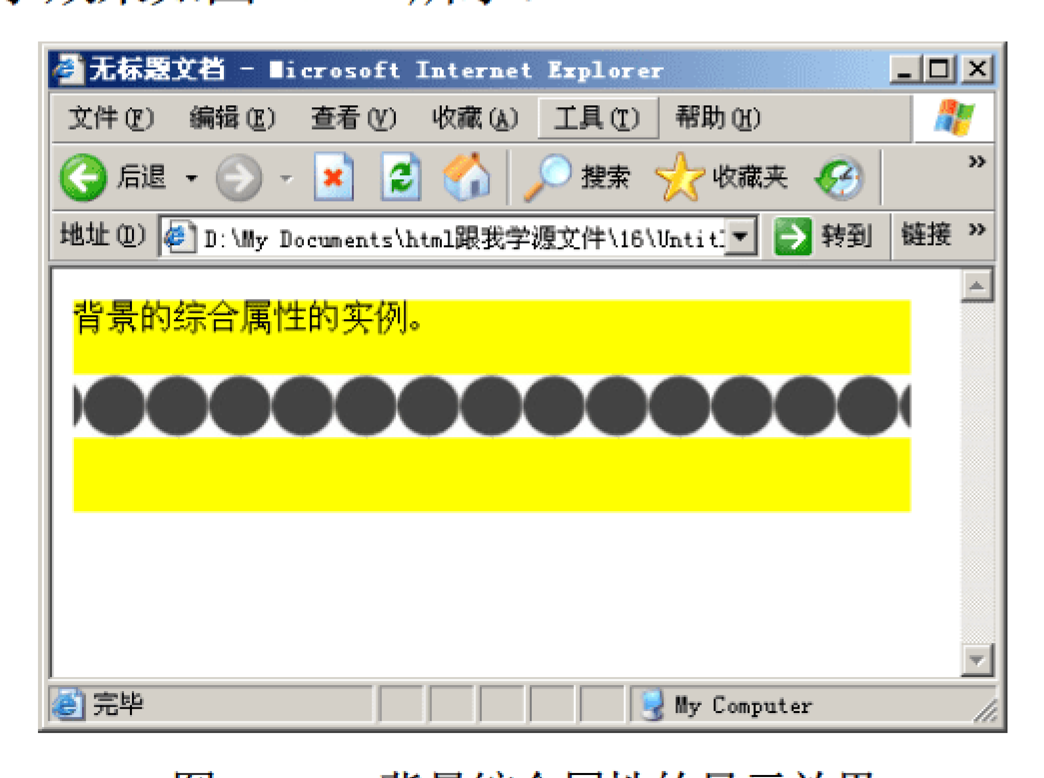


图 16-12　背景综合属性的显示效果

注意

在背景综合属性中，可以只定义部分背景属性。其他没有定义的背景属性，以默认的方式显示。

16.4 元素的补白

元素的补白部分用来分隔元素的内容部分和元素的边框部分。在 CSS 中，补白属性包括 4 个单侧补白属性和 1 个综合的属性，共 5 个属性。包括上补白属性(padding-top)、右补白属性(padding-right)、下补白属性(padding-bottom)、左补白属性(padding-left)和综合补白属性(padding)，下面分别进行讲解。

16.4.1 顶部补白属性 padding-top

顶部补白属性(padding-top)用来定义元素顶部边框与内容之间的距离。在顶部补白属性中，可以使用 3 种属性值，分别为长度值、百分比值和 auto 值。其语法结构如下所示。

```
padding-top:length | precent | auto;
```

下面是一个使用顶部补白属性的实例，其代码如下所示。

例程 16-11 padding-top.html

```
01 <html>
02 <head>
03 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
04 <title>无标题文档</title>
05 <style>
06 div{
   padding-top: 60px;
   background:#cccccc;
   width:300px;
   height:100px;
   }
07 </style>
08 </head>
09 <body>
   <div>顶部补白属性的实例。</div>
10 </body>
11 </html>
```

在以上的代码中，06 行定义元素的顶部补白属性为 60px，同时背景颜色属性值为浅灰色，元素内容大小为 300px×100px。此时元素中的内容与页面顶部之间将分隔 60px 的距离。以上代码的显示效果如图 16-13 所示。

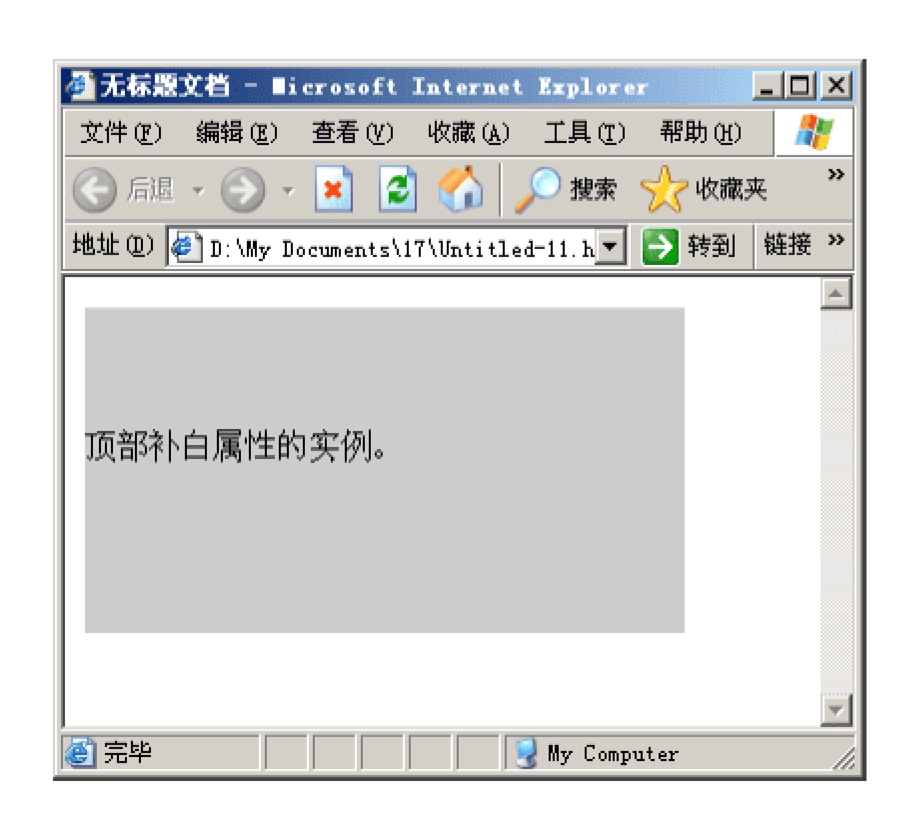


图 16-13 顶部补白属性的显示效果

16.4.2 右侧补白属性 padding-right

右侧补白属性(padding-right)用来定义元素右侧边框与内容之间的距离。在右侧补白属性中，可以使用 3 种属性值，分别为长度值、百分比值和 auto 值，其语法结构如下所示。

```
padding-right:length | precent | auto;
```

下面是一个使用右侧补白属性的实例，其代码如下所示。

例程 16-12 padding-right.html

```
01  <html >
02  <head>
03  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
04  <title>无标题文档</title>
05  <style>
06  div{
    padding-right:20%;
    background:yellow;
    width:300px;
    height:100px;
    }
07  </style>
08  </head>
09  <body>
    <div>注意观察元素中内容部分和元素背景右侧的距离。</div>
10  </body>
11  </html>
```

在以上的代码中，06 行定义元素的右侧补白属性为 20%，同时背景颜色属性值为黄色，元素内容大小为 300px×100px。此时元素中的内容与元素背景右侧之间将分隔整个页面 20%的距离(注意不是 300 的 20%)。以上代码的显示效果如图 16-14 所示。

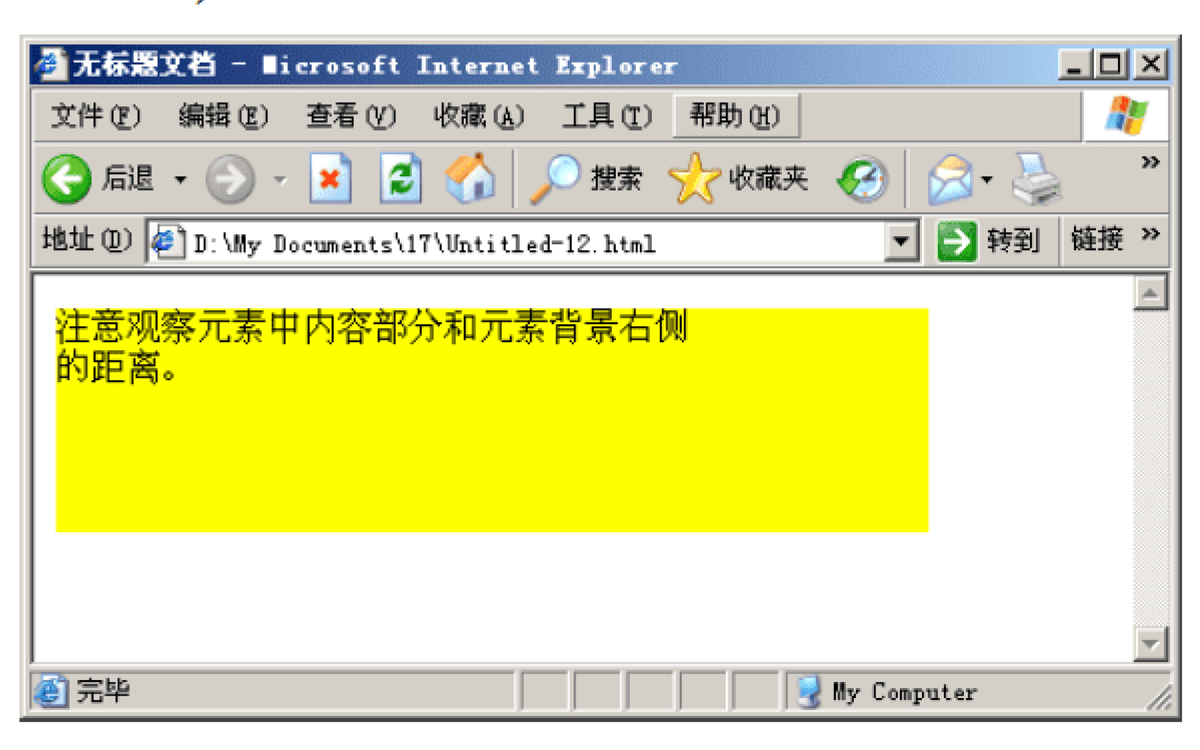


图 16-14　右侧补白属性的显示效果

16.4.3　底部补白属性 padding-bottom

底部补白属性(padding-bottom)用来定义元素顶部边框与内容之间的距离。在底部补白属性中，可以使用 3 种属性值，分别为长度值、百分比值和 auto 值，其语法结构如下所示。

```
padding-bottom:length | precent | auto;
```

下面是一个使用底部补白属性的实例，其代码如下所示。

例程 16-13 padding-bottom.html

```
<html >
<head>
```

```
03  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
04  <title>css 属性实例</title>
05  <style>
06  div{
    padding-bottom:100px;
    background:#999999;
    width:400px;
    height:100px;
  }
07  </style>
08  </head>
09  <body>
    <div>这是右侧补白属性的实例，注意观察的显示高度。</div>
10  </body>
11  </html>
```

在以上的代码中，06 行定义元素的底部补白属性为 100px，同时背景颜色属性值为黄色，元素内容大小为 400px×100px。此时元素中的显示高度(即元素背景部分)将增加 100px。以上代码的显示效果如图 16-15 所示。

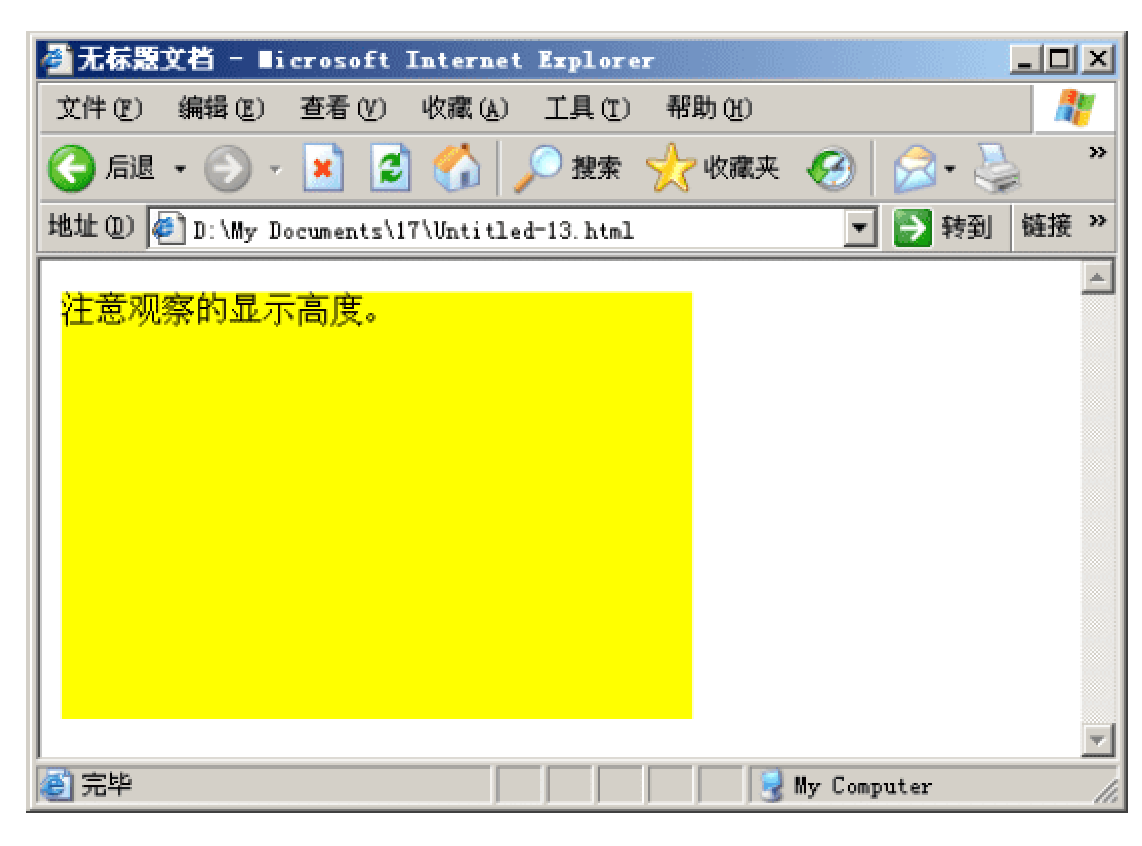


图 16-15　底部补白属性的显示效果

16.4.4　左侧补白属性 padding-left

左侧补白属性(padding-right)用来定义元素顶部边框与内容之间的距离。在左侧补白属性中，可以使用 3 种属性值，分别为长度值、百分比值和 auto 值，其语法结构如下所示。

```
padding-left:length | precent | auto;
```

下面是一个使用左侧补白属性的实例，其代码如下所示。

例程 16-14　padding-left.html

```
<html >
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>无标题文档</title>
```

```
05    <style>
06    div{
      padding-left:100px;
      background:yellow;
      width:300px;
      height:100px;
      }
07    </style>
08    </head>
09    <body>
      <div>注意观察元素内容与背景左侧边线之间的距离。</div>
10    </body>
11    </html>
```

在以上的代码中，06 行定义元素的左侧补白属性为 100px，同时背景颜色属性值为黄色，元素内容大小为 300px×100px。此时元素中的内容与元素背景左侧边线之间将分隔 100px 的距离。以上代码的显示效果如图 16-16 所示。

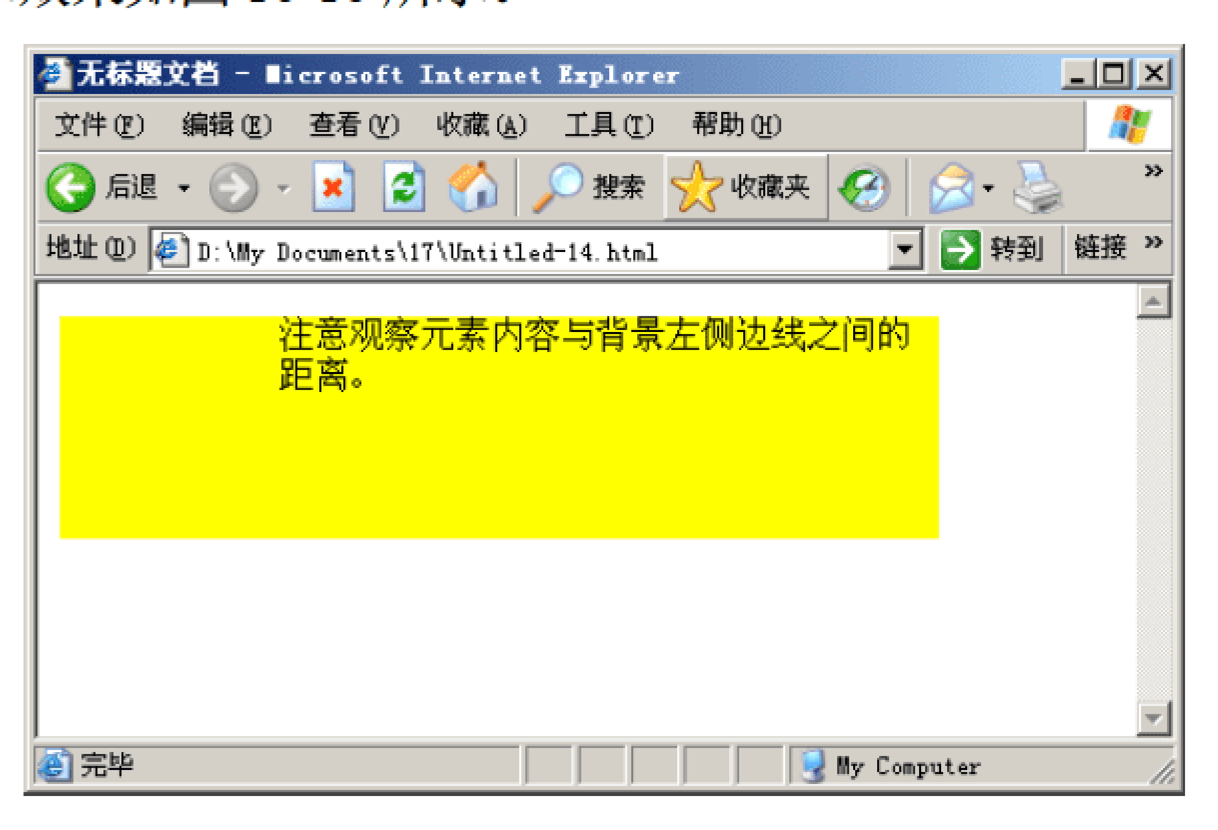


图 16-16　顶部补白属性的显示效果

16.4.5　综合补白属性 padding

综合补白属性(padding)用来统一定义元素的补白属性。在综合补白属性中，可以依次定义元素的四个补白属性，其语法结构如下所示。

```
padding: padding-top padding-right padding-bottom padding-left;
```

注意

在综合补白属性中，4 个单侧属性的顺序是固定的。4 个值的顺序，按照上、左、下、右的顺时针顺序依次排列。应用综合补白属性 padding 可以方便地设置页面，使用 padding 属性时，如果提供两个，第一个用于上－下，第二个用于左－右；如果提供 3 个，第一个用于上，第二个用于左－右，第三个用于下。

下面是一个使用综合补白属性的实例，其代码如下所示。

例程 16-15 padding.html

```
01 <html >
02 <head>
03 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
04 <title>无标题文档</title>
05 <style>
06 div{
   padding:20px 30px 40px 50px;
   background:yellow;
   width:150px;
   height:45px;
   }
07 </style>
08 </head>
09 <body>
   <div>这是个使用补白属性的实例，注意元素内容与元素四个边界之间的距离。</div>
10 </body>
11 </html>
```

06 行在以上的代码中，依次定义元素上、左、下、右 4 个边界的补白属性，分别为 20px、30px、40px、50px。以上代码的显示效果如图 16-17 所示。

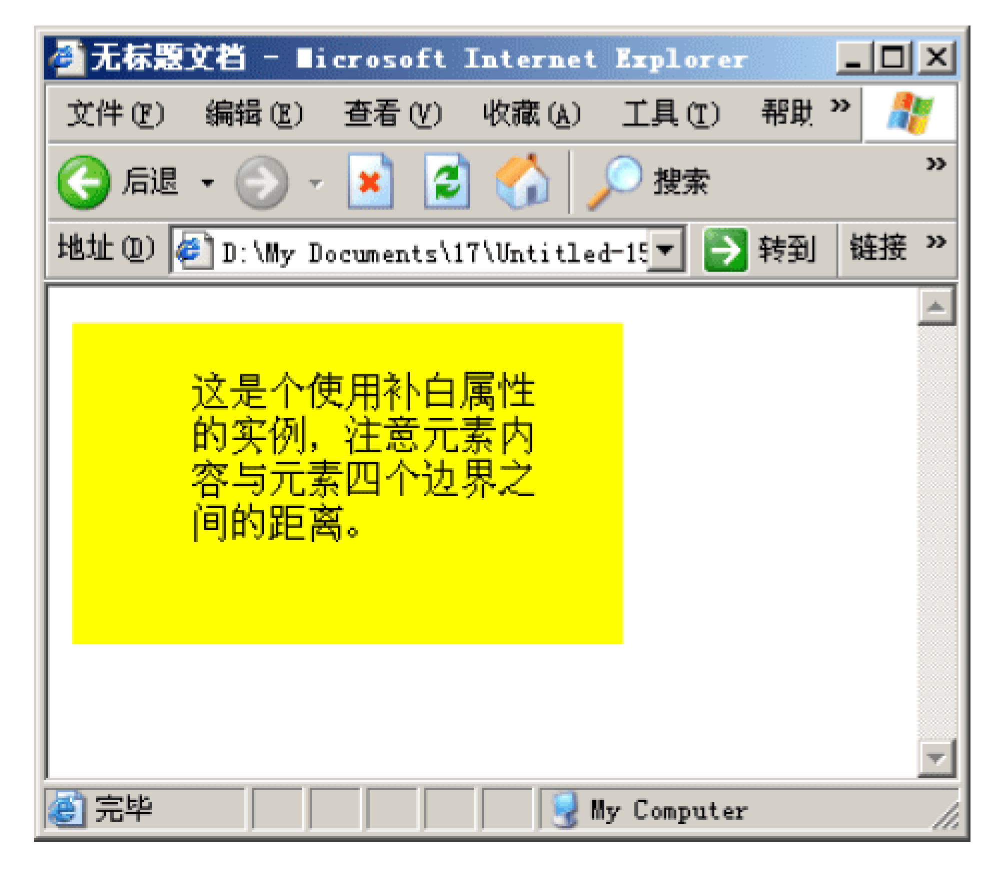


图 16-17 补白综合属性的显示效果

16.4.6 补白与背景

在元素的补白部分会显示元素定义的背景颜色或者背景图片。所以当定义补白属性的时候，同时也扩充了元素背景的显示范围。这在使用固定大小背景图片的时候非常重要。如果同时定义了元素的宽度、高度属性和补白属性，一定要计算背景图像的显示范围，以便能够完整地显示图像。

下面是一个关于补白与背景属性的实例，其代码如下所示。

例程 16-16 example.html

```
<html >
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>无标题文档</title>
<style>
div{
padding:60px 60px;
background:url(images/pic.jpg) no-repeat top left;
width:100px;
height:84px;
```

```
       }
07     </style>
08     </head>
09     <body>
       <div>这是个使用补白属性的实例，注意元素内容与元素四个边界之间的距离。</div>
10     </body>
11     </html>
```

在以上的代码中 06 行通过定义补白和背景属性显示了背景图片，其显示效果如图 16-18 所示。如果更改补白属性的属性值，则背景图片无法正常显示。将补白属性的值均定义为 60px，则页面的显示效果如图 16-19 所示。



图 16-18　补白属性中显示背景的效果

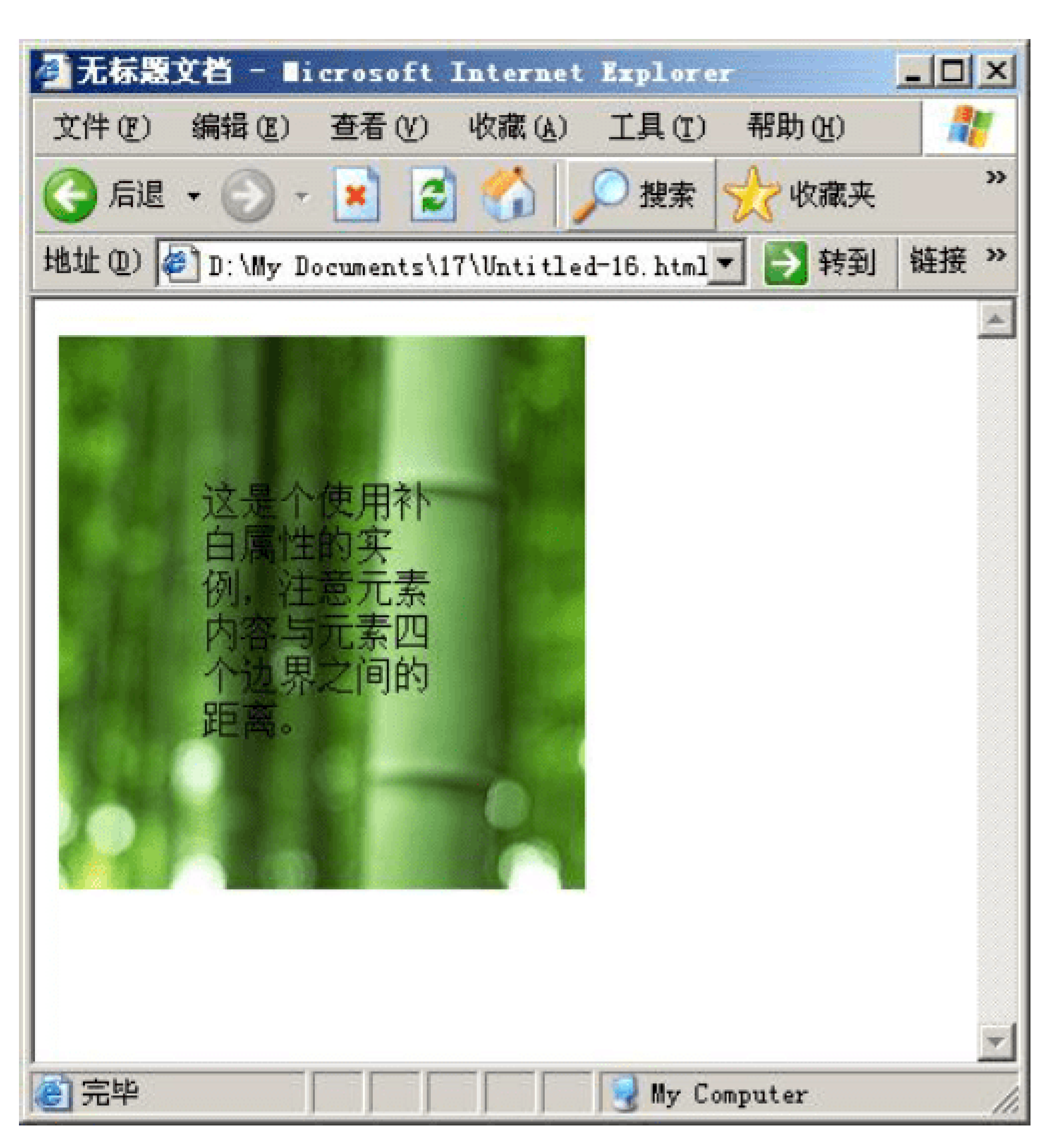


图 16-19　更改补白属性值的显示效果

16.5 元素的边框

元素的边框，一般用来起到分隔页面各个部分的效果。在 CSS 中，边框属性可以分为以下几个属性。包括边框样式(border-style)、边框宽度(border-width)、边框颜色(border-color)、边框样式(border-style)以及边框综合属性(border)等。

在 CSS 中，可以分别定义每一侧边框的样式。在每一侧的边框样式中，同样可以定义边框的样式、宽度、颜色等，下面分别进行讲解。

16.5.1 顶部边框样式属性 border-top-style

顶部边框样式属性(border-top-style)用来定义元素顶部边框的显示效果。在顶部边框样式属性中，可以使用的取值有很多，其中有部分取值还不被浏览器所支持。其语法结构如下所示。

```
border-top-style: none | hidden | dotted | dashed | solid | double | groove | ridge | inset | outset;
```

其中各个属性的含义如下所示。

- none：没有边框。
- hidden：隐藏边框。
- dotted：边框为点线。
- dashed：边框为虚线。
- solid：边框为实线。
- double：边框为双线。
- groove：边框为 3d 凹槽。
- ridge：边框为菱形。
- inset：边框为 3d 凹边。
- outset：边框为 3d 凸边。

注意

以上的几个属性中，groove、ridge、inset、outset4 个属性值的显示效果，要受到边框颜色属性(border-color)的影响，IE 浏览器并不支持这 4 个属性值。同时 IE 浏览器也不支持 hidden 属性值。

下面是一个使用顶部边框样式属性的实例，其代码如下所示。

例程 16-17 border-top-style.html

```
<html >
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>无标题文档</title>
<style>
.dashed{
border-top-style:dashed;
}
.double{
border-top-style: double;}
.ridge{
border-top-style: ridge;
}
div{
border-top-color:#000000;
border-top-width:5px;
width:400px;
height:30px;
background:yellow;
}
</style>
```

```
11   </head>
12   <body>
     <div class="dashed"></div>
     <div class="double"></div>
     <div class="ridge"></div>
     </body>
13   </html>
```

在以上的代码中，06 到 08 行分别定义了 3 种不同的顶部边框样式。同时 09 行定义了元素的宽度、高度，以及边框的颜色和宽度属性，用来显示元素边框样式。以上代码的显示效果如图 16-20 所示。

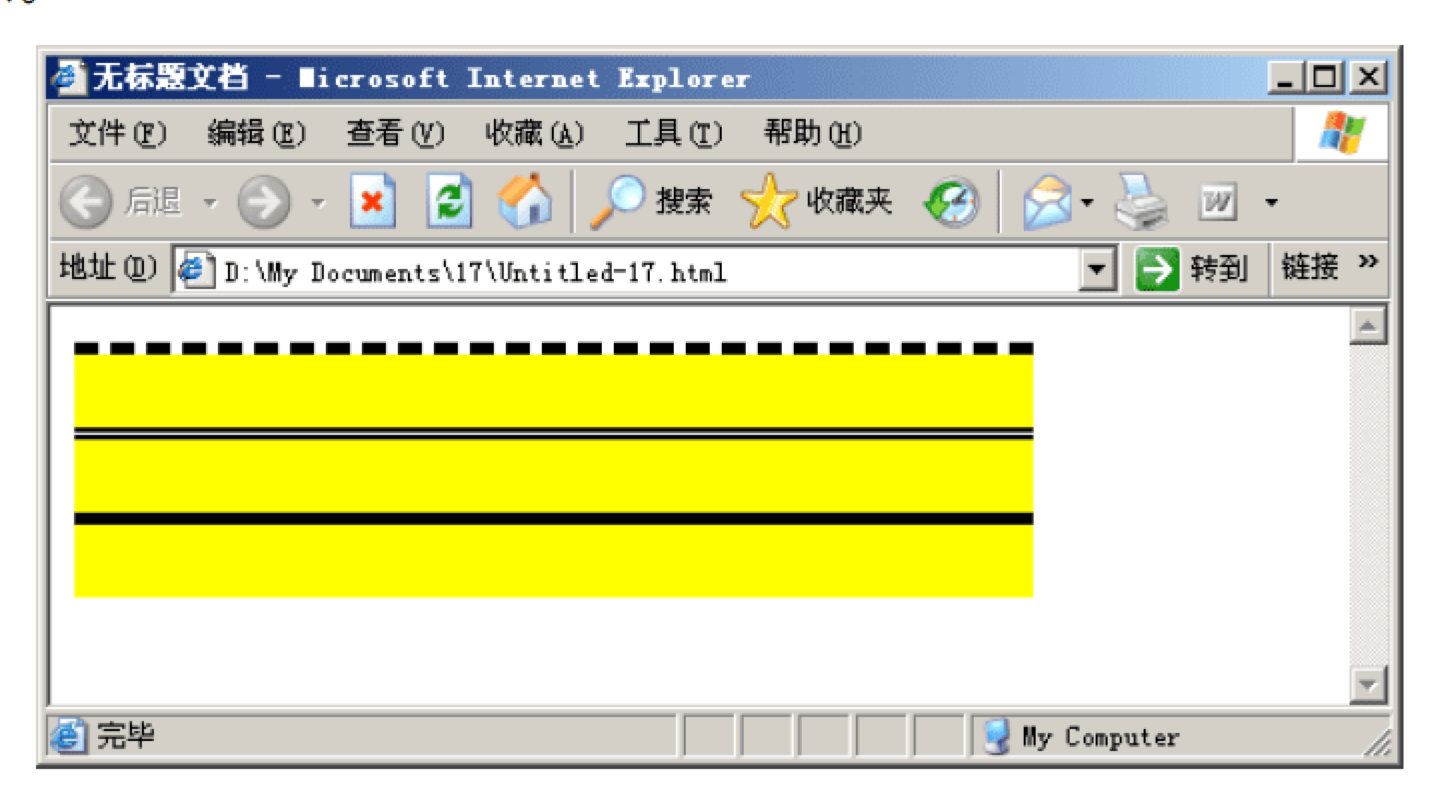


图 16-20　顶部补白属性的显示效果

注意

如果要显示边框的样式，就必须要定义边框的颜色属性和宽度属性。否则元素的边框无法显示。

16.5.2　右侧边框样式属性 border-right-style

右侧边框样式属性(border-right-style)用来定义元素右侧边框的显示效果。在右侧边框样式属性中，可以使用的取值有很多，其中有部分取值还不被浏览器所支持。其语法结构如下所示。

```
border-top-style: none | hidden | dotted | dashed | solid | double | groove | ridge | inset | outset;
```

其中各个属性值的含义参见 16.5.1 小节的相关内容。

下面是一个使用右侧边框样式属性的实例，其代码如下所示。

例程 16-18　border-right-style.html

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>无标题文档</title>
```

```
05  <style>
06  .solid{
    border-right-style:solid;
    }
07  .double{
    border-right-style:double;
    }
08  div{
    float:left;
    border-right-color:#000000;
    border-top-width:6px;
    width:80px;
    height:150px;
    background:#999999
  }
09  </style>
10  </head>
11  <body>
    <div class="solid"></div>
    <div class="double"></div>
12  </body>
13  </html>
```

在以上的代码中，06 和 07 行分别定义了两种不同的右侧边框样式。同时 08 行定义了元素的宽度值小于高度值，以及边框的颜色、宽度属性和浮动属性，其目的是用来显示元素边框样式。以上代码的显示效果如图 16-21 所示。

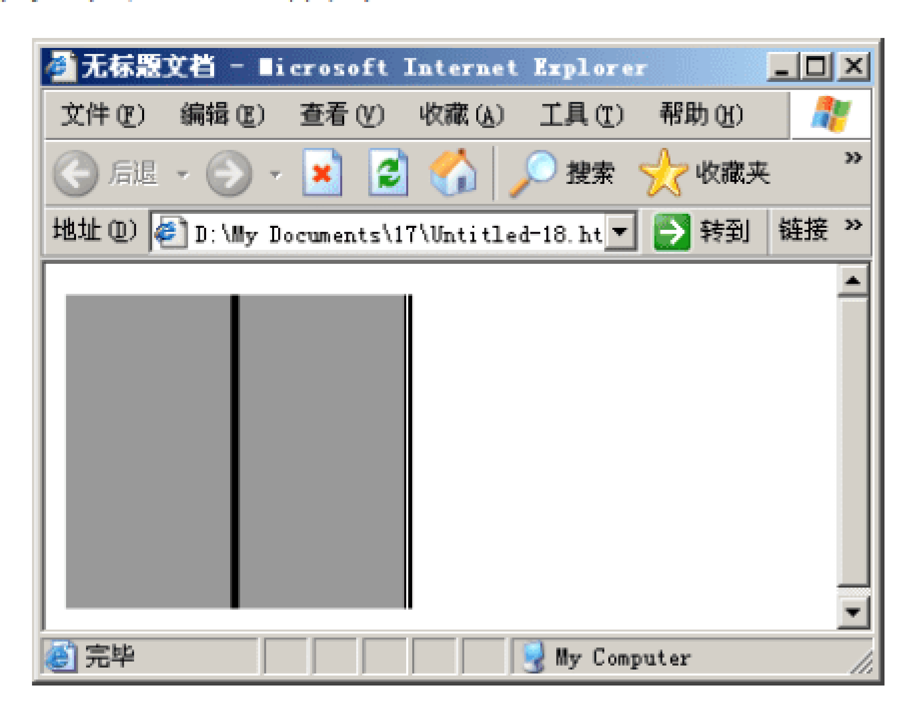


图 16-21　右侧边框属性的显示效果

16.5.3　底部边框样式属性 border-bottom-style

底部边框样式属性(border-bottom-style)用来定义元素底部边框的显示效果。在底部边框样式属性中，可以使用的取值有很多，其中有部分取值还不被浏览器所支持。其语法结构如下所示。

```
border-top-style: none | hidden | dotted | dashed | solid | double | groove | ridge | inset | outset;
```

其中各个属性值的含义参见 16.5.1 小节的相关内容。

下面是一个使用底部边框样式属性的实例，其代码如下所示。

例程 16-19 border-bottom-style.html

```
01  <html >
02  <head>
03  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
04  <title>无标题文档</title>
05  <style>
06  .dotted{
    border-bottom-style:dotted;}
07  .double{
    border-bottom-style:double;
    }
08  div{
    border-bottom-color:#000000;
    border-top-width:4px;
    width:400px;
    height:40px;
    }
09  </style>
10  </head>
11  <body>
    <div class="dotted"></div>
    <div class="double"></div>
12  </body>
13  </html>
```

在以上的代码中，06 和 07 行分别定义了两种不同的底部边框样式。同时 08 行定义了元素的宽度、高度，以及边框的颜色和宽度属性，其目的是用来显示元素边框样式。以上代码的显示效果如图 16-22 所示。

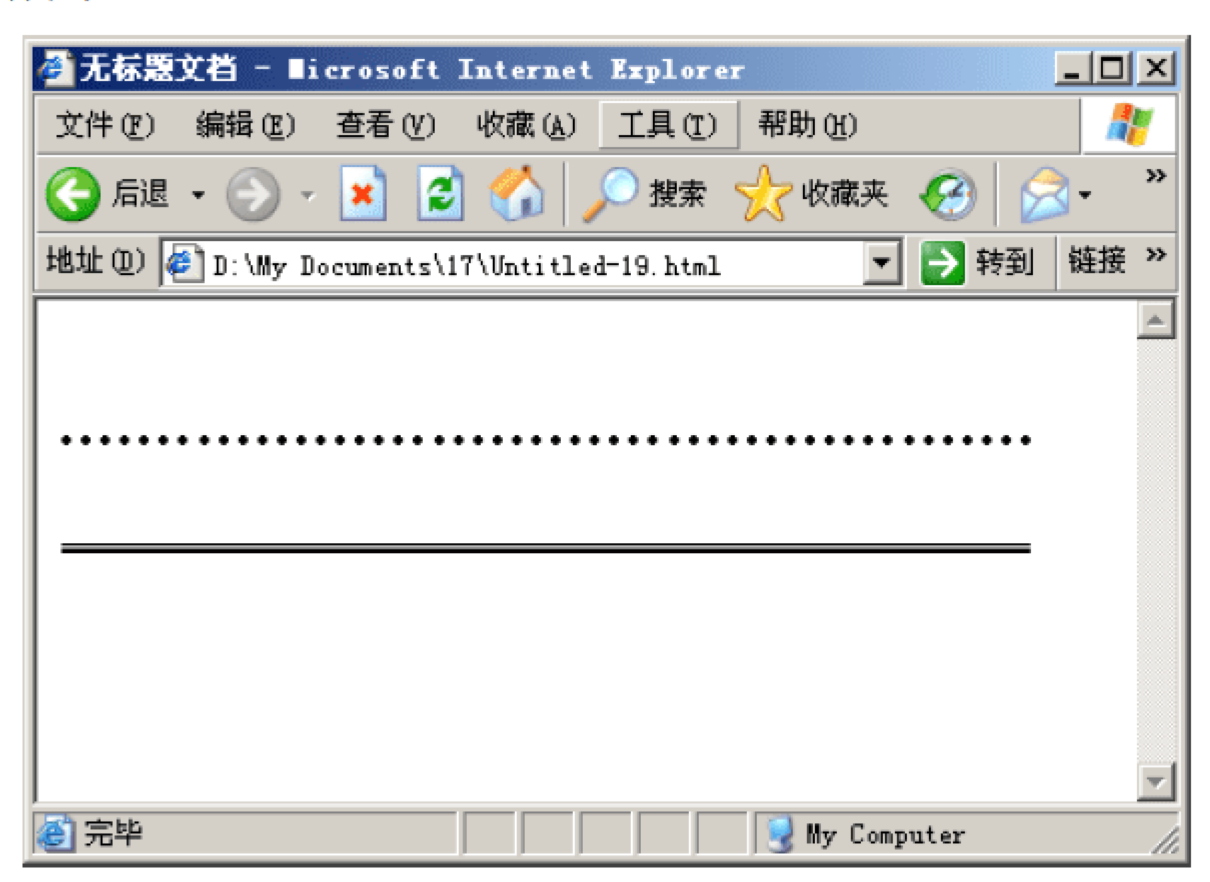


图 16-22 底部边框样式属性的显示效果

16.5.4 左侧边框样式属性 border-left-style

左侧边框样式属性(border-left-style)用来定义元素左侧边框的显示效果。在左侧边框样式属性中，可以使用的取值有很多，其中有部分取值还不被浏览器所支持。其语法结构如下所示。

```
border-top-style: none | hidden | dotted | dashed | solid | double | groove | ridge | inset | outset;
```

其中各个属性值的含义参见 16.5.1 小节的相关内容。

下面是一个使用左侧边框样式属性的实例，其代码如下所示。

例程 16-20 border-left-style.html

```
01 <html>
02 <head>
03 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
04 <title>无标题文档</title>
05 <style>
06 .dashed{
   border-left-style:dashed;
   }
07 .solid{
   border-left-style:solid;
   }
08 .double{
   border-left-style:double;
   }
09 div{
   float:left;
   border-left-color:#000000;
   border-top-width:5px;
   width:80px;
   height:150px;
   }
10 </style>
11 </head>
12 <body>
   <div class="dashed"></div>
   <div class="solid"></div>
   <div class="double"></div>
13 </body>
14 </html>
```

在以上的代码中，06 到 08 行分别定义了 3 种不同的单侧边框样式。同时 09 行定义了元素的宽度、高度，以及边框的颜色、宽度属性和浮动属性，其目的是用来显示元素边框样式。以上代码的显示效果如图 16-23 所示。

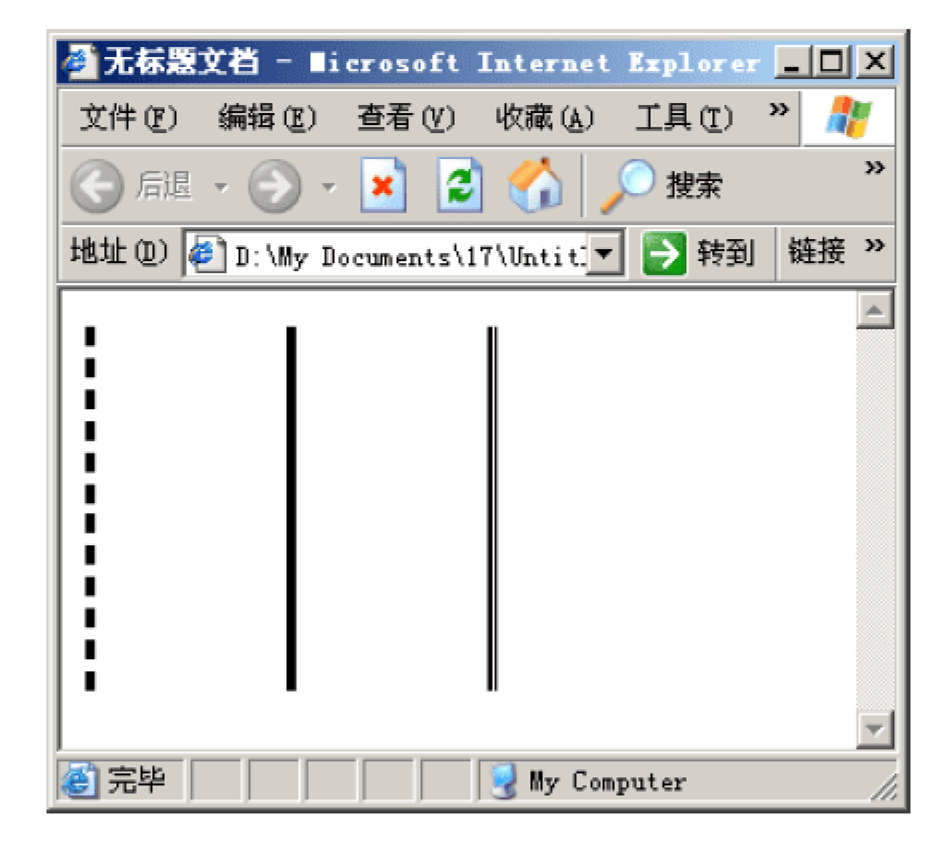


图 16-23　左侧补白属性的显示效果

16.5.5　顶部边框颜色属性 border-top-color

顶部边框颜色属性(border-top-color)用来定义元素顶部边框的颜色。顶部边框颜色属性的取值为颜色值，其语法结构如下所示。

```
border-top-color: color | colorname;
```

注意

边框颜色属性的默认值为黑色。当未定义任何边框颜色属性的时候，边框显示黑色。

下面是一个使用顶部边框颜色属性的实例，其代码如下所示。

例程 16-21　border-top-color.html

```
<html >
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>css 属性实例</title>
<title>无标题文档</title>
<style>
div{
border-top-color:yellow;
border-top-style:solid;
border-top-width:4px;
width:400px;
height:80px;
background:#cccccc
}
</style>
</head>
```

```
10  <body>
    <div></div>
11  </body>
12  </html>
```

在以上的代码中，07 行定义了顶部边框的颜色为黄色。同时定义了一个 400px×80px 的元素，以及边框的样式和宽度属性，目的是用来显示元素边框颜色。以上代码的显示效果如图 16-24 所示。

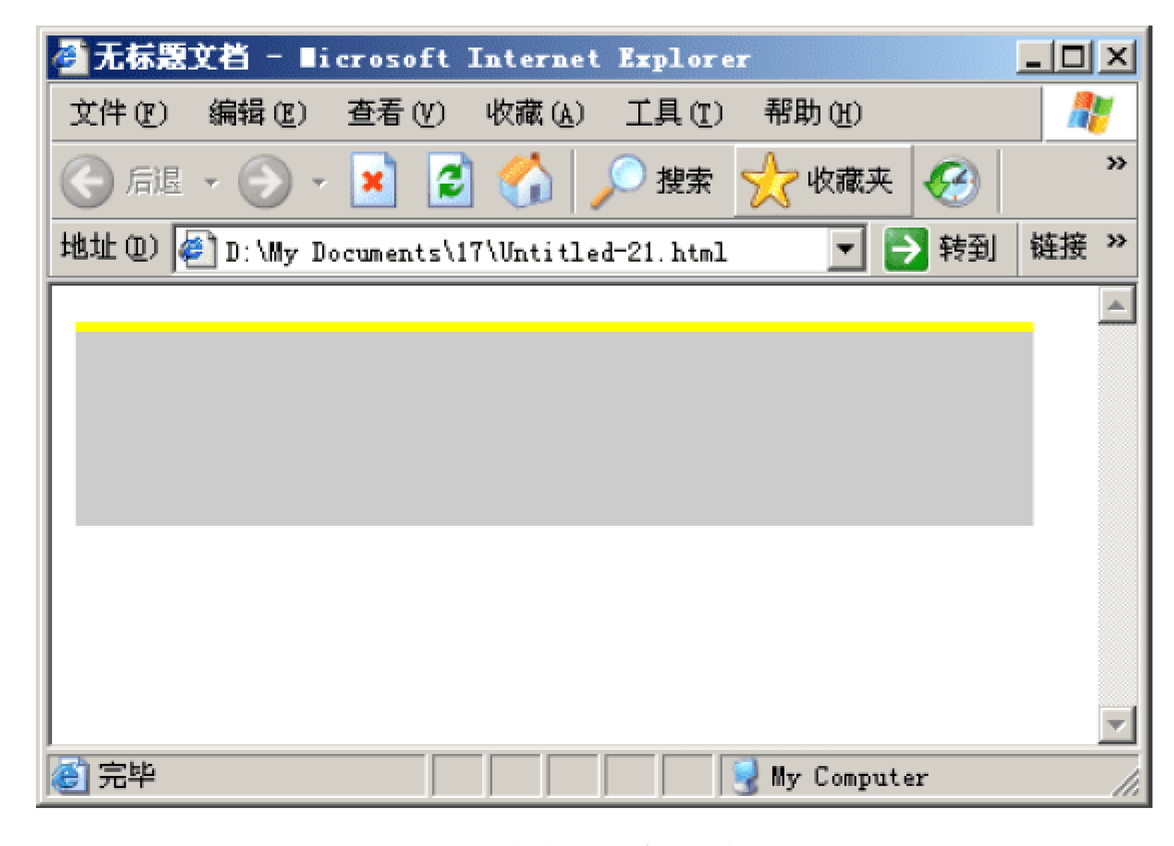


图 16-24　顶部边框颜色属性的显示效果

16.5.6　右侧边框颜色属性 border-right-color

右侧边框颜色属性(border-right-color)用来定义元素右侧边框的颜色。右侧边框颜色属性的取值为颜色值，其语法结构如下所示。

```
border-right-color: color | colorname;
```

下面是一个使用右侧边框颜色属性的实例，其代码如下所示。

例程 16-22　border-right-color.html

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>无标题文档</title>
<style>
div{
border-right-color: #00FF00;
border-right-style:solid;
border-right-width:40px;
width:300px;
height:120px;
}
</style>
</head>
<body>
```

```
      <div></div>
10    </body>
11    </html>
```

在以上的代码中，06 行定义了右侧边框的颜色为绿色。同时定义了元素的宽度、高度，以及边框的样式和宽度属性，其目的是用来显示元素边框颜色。以上代码的显示效果如图 16-25 所示。

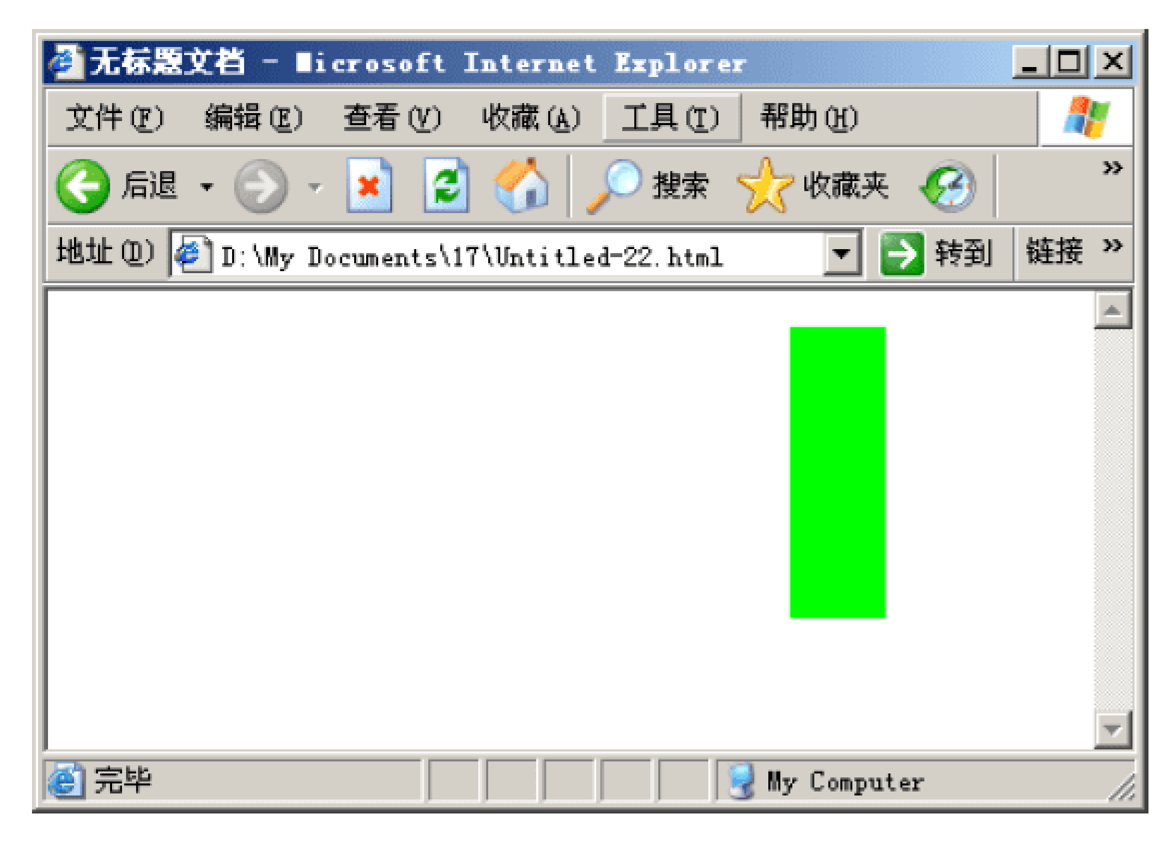


图 16-25　右侧边框颜色属性的显示效果

16.5.7　底部边框颜色属性 border-bottom-color

底部边框颜色属性(border-bottom-color)用来定义元素底部边框的颜色。底部边框颜色属性的取值为颜色值，其语法结构如下所示。

```
border-bottom-color: color | colorname;
```

下面是一个使用底部边框颜色属性的实例，其代码如下所示。

例程 16-23　border-bottom-color.html

```
01    <html >
02    <head>
03    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
04    <title>无标题文档</title>
05    <style>
06    div{
      border-bottom-color: #FFFF00;
      border-bottom-style:solid;
      border-bottom-width:50px;
      width:300px;
      height:150px;
    }
07    </style>
08    </head>
09    <body>
      <div></div>
```

```
10  </body>
11  </html>
```

在以上的代码中，06 行定义了底部边框的颜色为黄色。同时定义了元素的宽度、高度，以及边框的样式和宽度属性，其目的是用来显示元素边框颜色。以上代码的显示效果如图 16-26 所示。

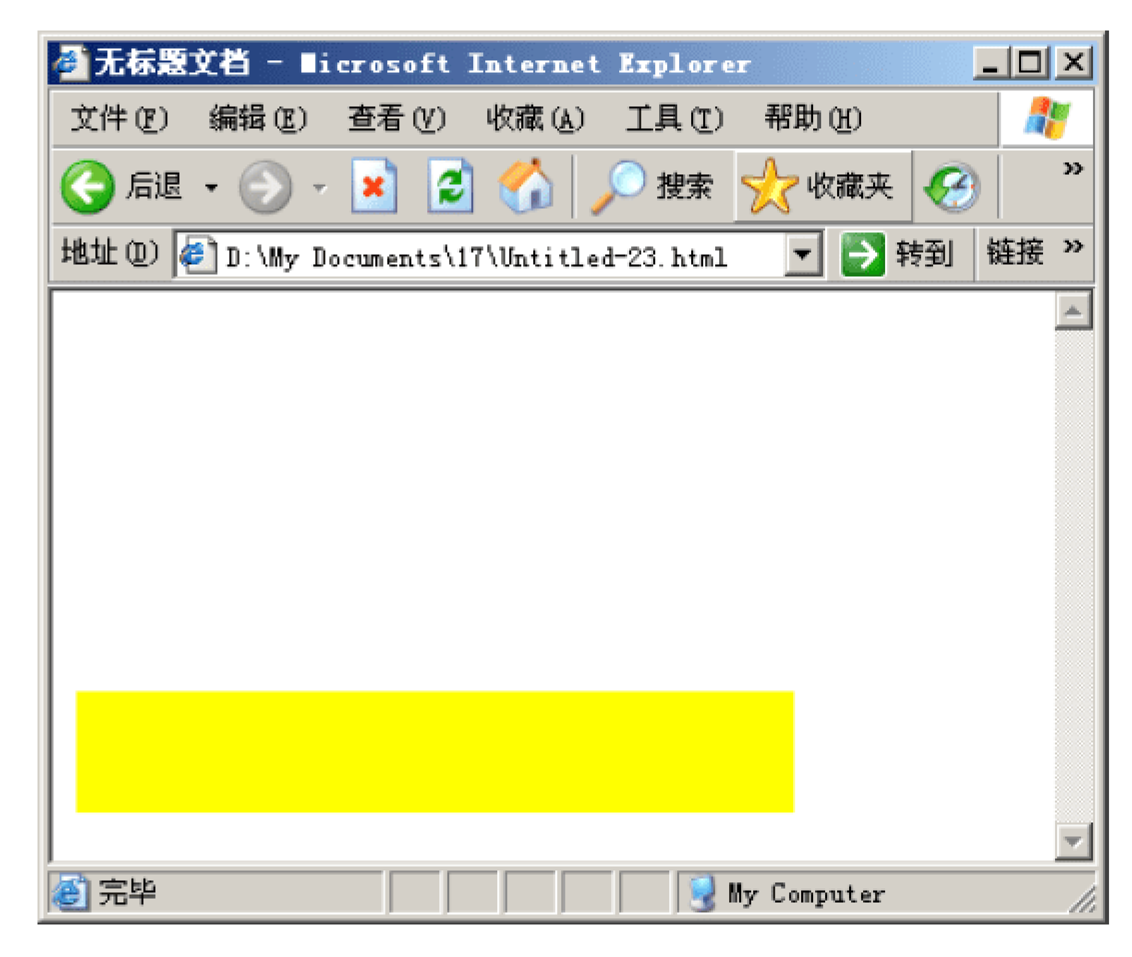


图 16-26　底部边框颜色属性的显示效果

16.5.8　左侧边框颜色属性 border-left-color

左侧边框颜色属性(border-left-color)用来定义元素左侧边框的颜色。左侧边框颜色属性的取值为颜色值，其语法结构如下所示。

```
border-left-color: color | colorname;
```

下面是一个使用左侧边框颜色属性的实例，其代码如下所示。

例程 16-24　border-left-color.html

```
<html >
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>无标题文档</title>
<style>
div{
border-left-color: #FFFF00;
border-left-style:solid;
border-left-width:40px;
width:200px;
height:150px;
}
</style>
</head>
<body>
```

```
    <div></div>
10  </body>
11  </html>
```

在以上的代码中，06 行定义了左侧边框的颜色为黄色。同时定义了元素的宽度、高度，以及边框的样式和宽度属性，其目的是用来显示元素边框颜色。以上代码的显示效果如图 16-27 所示。

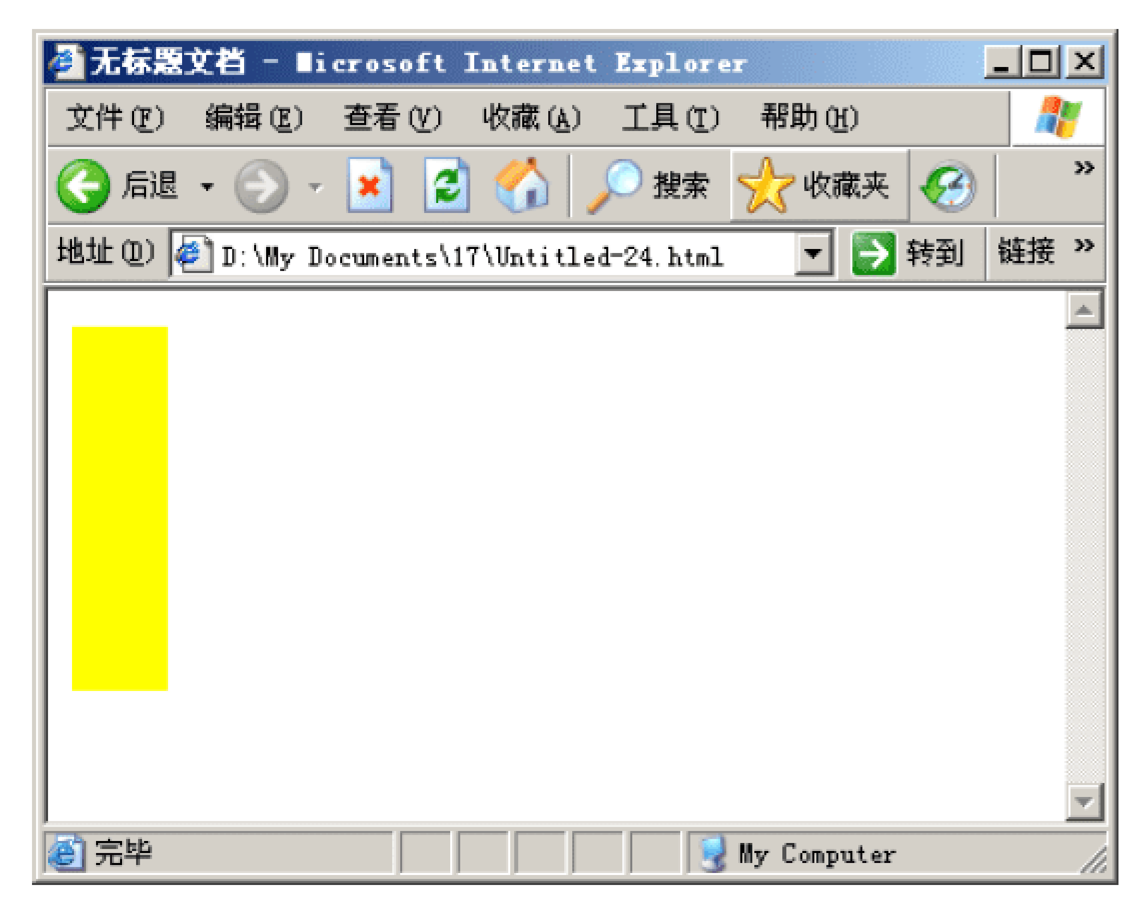


图 16-27　左侧边框颜色属性的显示效果

16.5.9　顶部边框宽度属性 border-top-width

顶部边框宽度属性(border-top-width)用来定义元素顶部边框的宽度。顶部边框宽度属性中使用的属性值为长度值以及 medium、thin、thick 等，其语法结构如下所示。

```
border-top-width: medium | thin | thick | length;
```

其中各个属性的含义如下所示。

- medium：默认的边框宽度。
- thin：小于默认的边框宽度。
- thick：边框将使用加粗效果。
- length：长度值。

下面是一个使用顶部边框宽度属性的实例，其代码如下所示。

例程 16-25　border-top-width.html

```
<html >
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>无标题文档</title>
<style>
div{
border-top-width: thick;
border-top-style:solid;
```

```
    width:300px;
    height:80px;
     }
07  </style>
08  </head>
09  <body>
    <div></div>
10  </body>
11  </html>
```

在以上的代码中，06 行定义顶部边框宽度值为 thick，此时边框将使用加粗效果。以上代码的显示效果如图 16-28 所示。

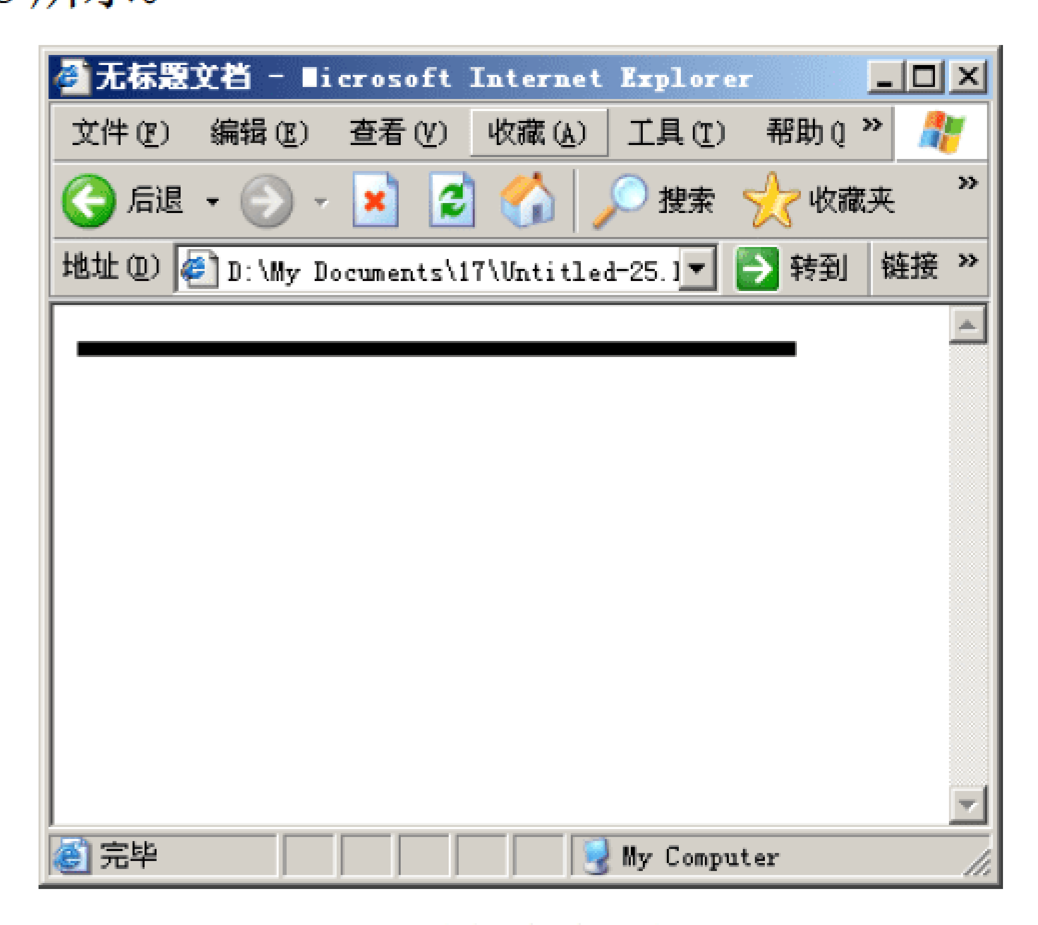


图 16-28 顶部边框宽度属性的显示效果

16.5.10 右侧边框宽度属性 border-right-width

右侧边框宽度属性(border-right-width)用来定义元素右侧边框的宽度。右侧边框宽度属性中使用的属性值为长度值以及 medium、thin、thick 等，其语法结构如下所示。

```
border-right-width: medium | thin | thick | length;
```

其中各个属性的含义参见 16.5.9 小节的相关内容。

下面是一个使用右侧边框宽度属性的实例，其代码如下所示。

例程 16-26 border-right-width.html

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>无标题文档</title>
<style>
div{
border-right-width:thin;
border-right-style:solid;
width:120px;
```

```
        height:180px;
        }
07      </style>
08      </head>
09      <body>
        <div></div>
10      </body>
11      </html>
```

在以上的代码中，06 行定义顶部边框宽度值为 thin，此时边框将显示细线的效果。以上代码的显示效果如图 16-29 所示。

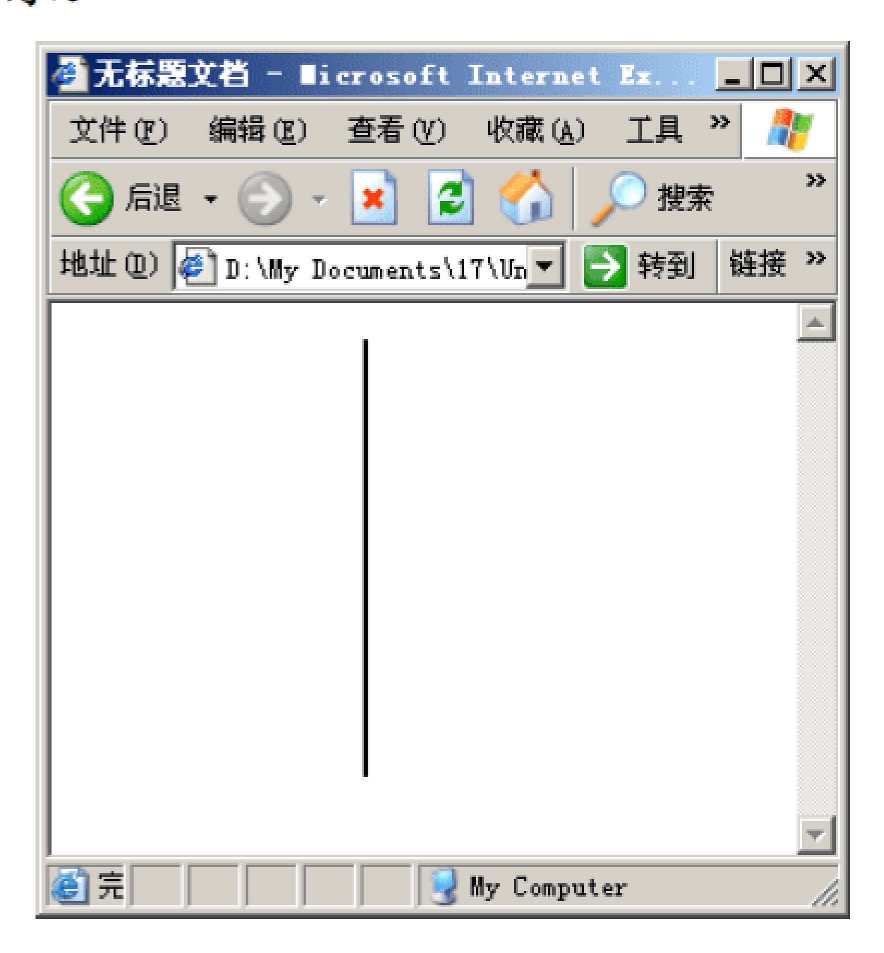


图 16-29　右侧边框宽度属性的显示效果

16.5.11　底部边框宽度属性 border-bottom-width

底部边框宽度属性(border-bottom-width)用来定义元素底部边框的宽度。底部边框宽度属性中使用的属性值为长度值以及 medium、thin、thick 等，其语法结构如下所示。

```
border-bottom-width: medium | thin | thick | length;
```

其中各个属性的含义参见 16.5.9 小节的相关内容。

下面是一个使用底部边框宽度属性的实例，其代码如下所示。

例程 16-27　border-bottom-width.html

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>无标题文档</title>
<style>
div{
border-bottom-width: medium;
border-bottom-style:solid;
width:300px;
```

```
        height:200px;
        }
07    </style>
08    </head>
09    <body>
        <div></div>
10    </body>
11    </html>
```

在以上的代码中，06 行定义底部边框宽度值为默认值显示，通常状况下实际显示的值为 5px。以上代码的显示效果如图 16-30 所示。

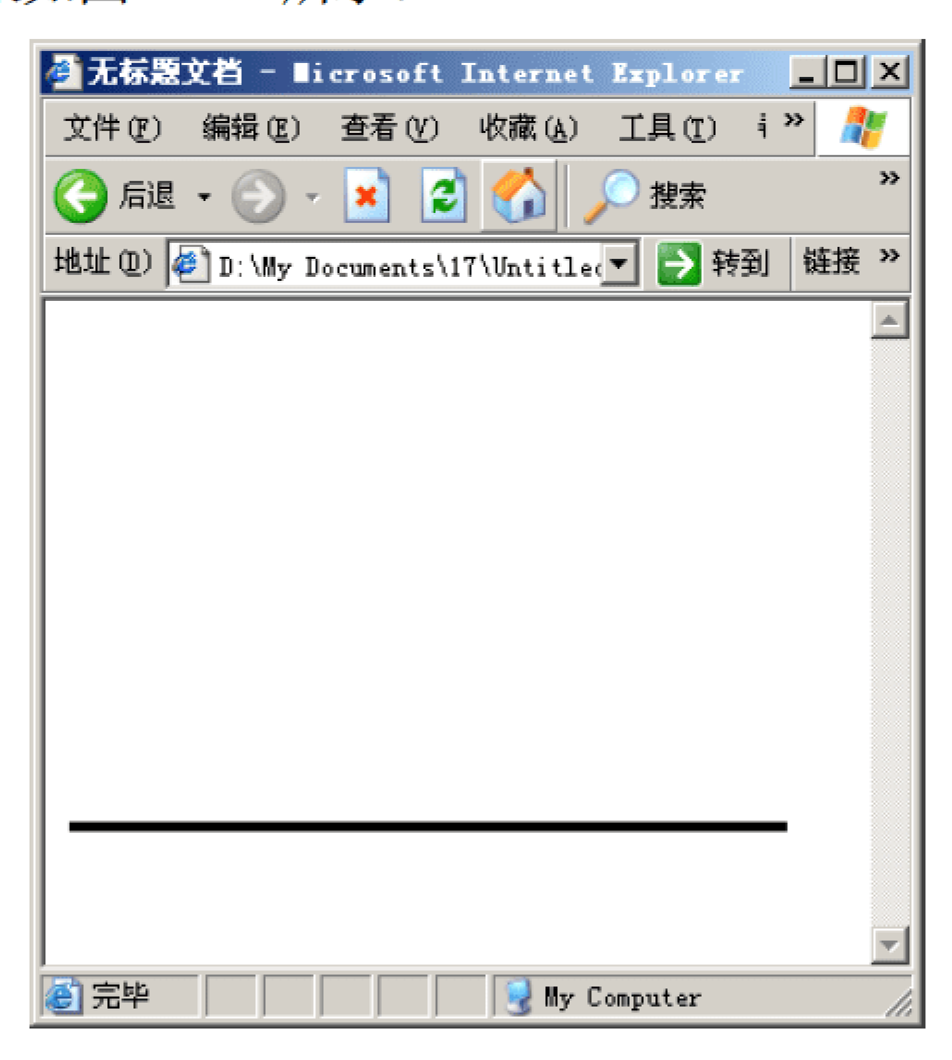


图 16-30　底部边框宽度属性的显示效果

16.5.12　左侧边框宽度属性 border-left-width

左侧边框宽度属性(border-left-width)用来定义元素左侧边框的宽度。左侧边框宽度属性中使用的属性值为长度值以及 medium、thin、thick 等，其语法结构如下所示。

```
border-left-width: medium | thin | thick | length;
```

其中各个属性的含义参见 16.5.9 小节的相关内容。

下面是一个使用左侧边框宽度属性的实例，其代码如下所示。

例程 16-28　border-left-width.html

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>无标题文档</title>
<style>
div{
    border-left-width:40px;
    border-left-style:groove;
```

```
        width:120px;
        height:180px;
        }
07  </style>
08  </head>
09  <body>
        <div></div>
10  </body>
11  </html>
```

在以上的代码中，06 行定义左侧边框宽度值为 40px。以上代码的显示效果如图 16-31 所示。

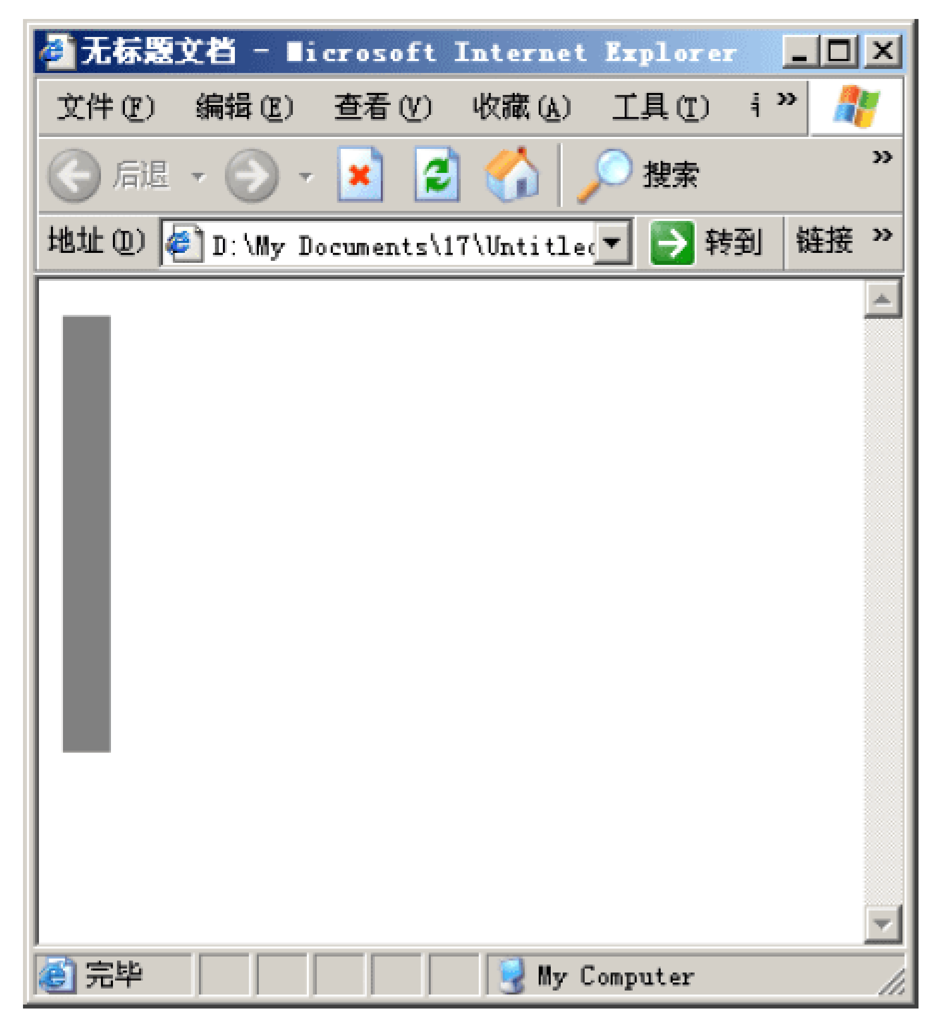
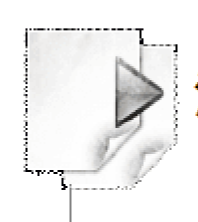


图 16-31　左侧边框宽度属性的显示效果

16.5.13　边框样式属性 border-style

边框样式属性(border-style)用来统一定义边框的显示样式。在边框样式属性中，使用的属性值和相应的边框单侧样式属性中相同。



注意

使用边框样式属性，无法定义某一个方向上边框显示独立的样式效果。

下面是一个使用边框样式属性的实例，其代码如下所示。

例程 16-29　border-style.html

```
<html >
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>无标题文档</title>
<style>
```

```
06  div{
    border-style: dotted;
    width:300px;
    height:100px;
  }
07  </style>
08  </head>
09  <body>
    <div></div>
10  </body>
11  </html>
```

在以上的代码中，06 行定义边框样式为点线。代码运行后的显示效果如图 16-32 所示。

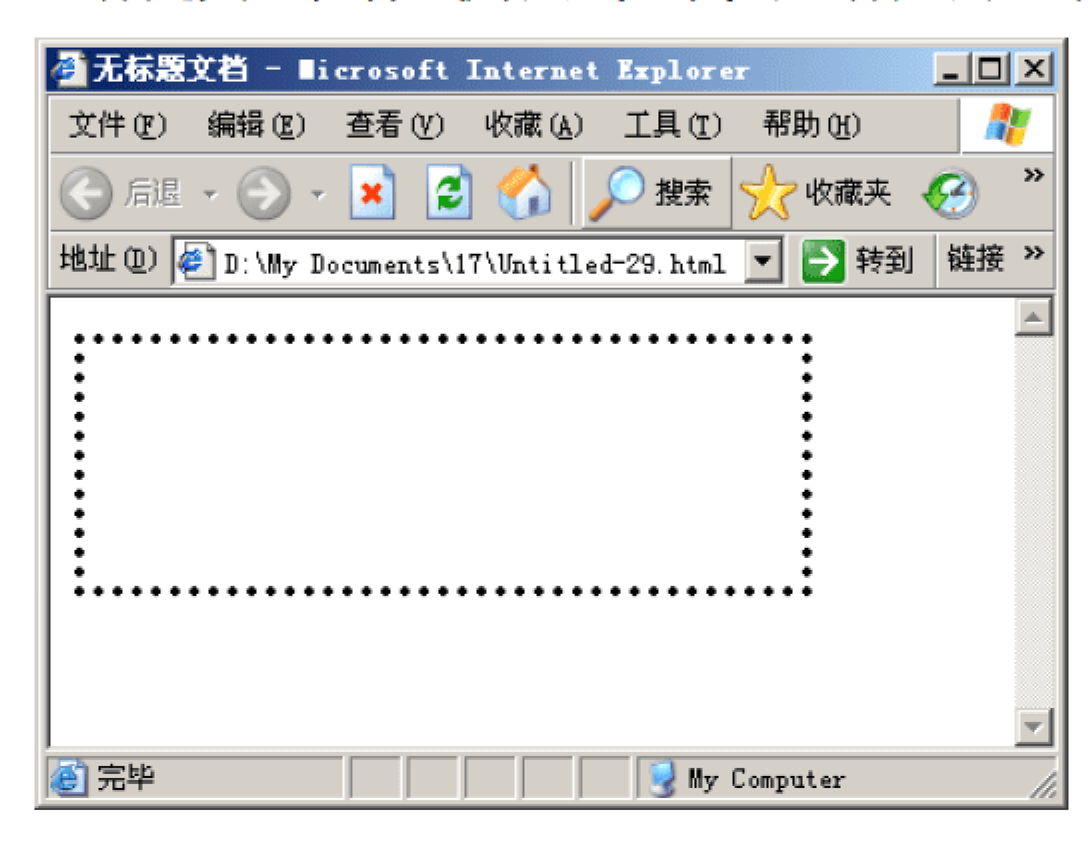
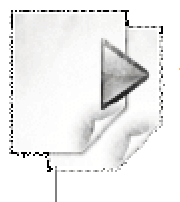


图 16-32　边框样式属性的显示效果

16.5.14　边框颜色属性 border-color

边框颜色属性(border-color)用来统一定义边框的显示颜色。在边框颜色属性中，使用的属性值为颜色值。



注意

使用边框颜色属性，无法定义某一个方向上边框显示独立的颜色效果。

下面是一个使用边框颜色属性的实例，其代码如下所示。

例程 16-30　border-color.html html.html

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>无标题文档</title>
<style>
div{
border-color: #FFFF33;
```

```
        border-width:20px;
        border-style:dotted;
        width:300px;
        height:150px;
    }
07  </style>
08  </head>
09  <body>
    <div></div>
10  </body>
11  </html>
```

在以上的代码中，06 行定义边框颜色为黄色，边框宽度为 20 像素，边框样式为点线。代码运行后的显示效果如图 16-33 所示。

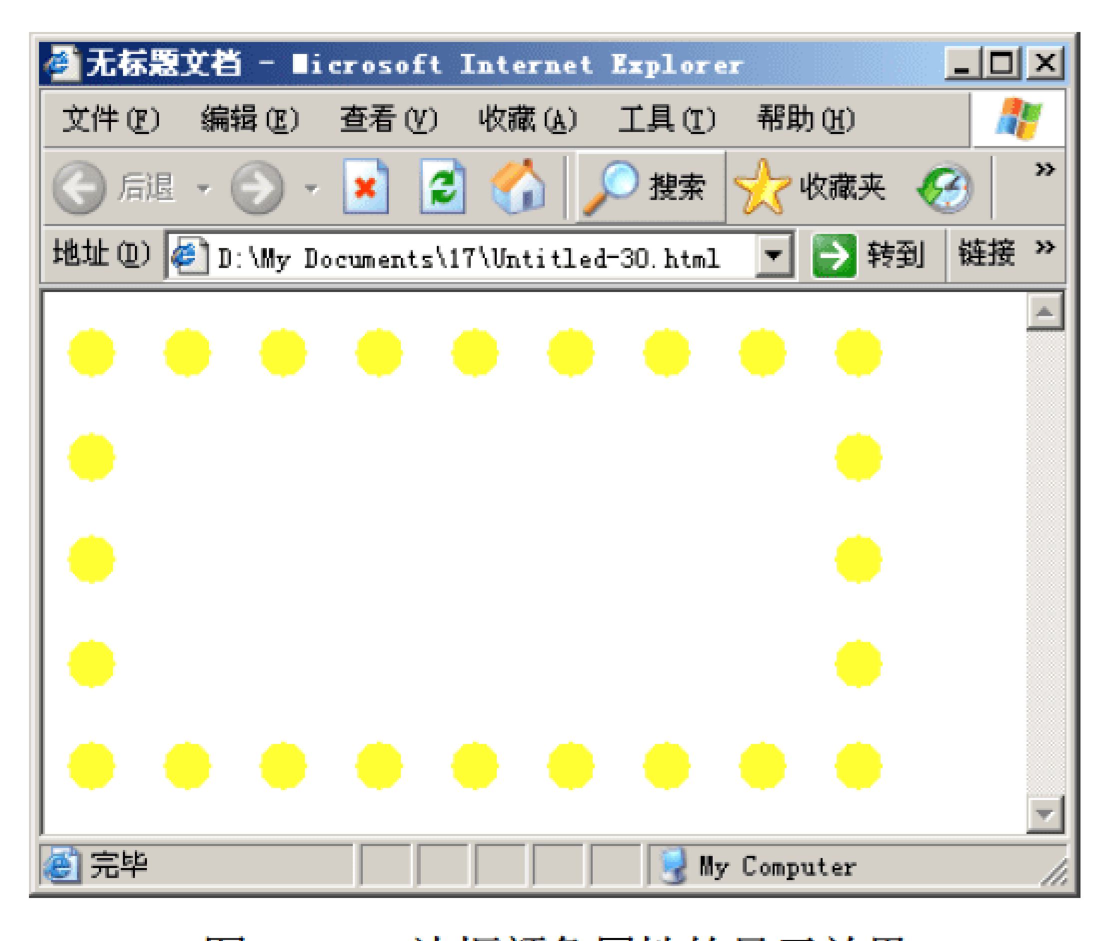
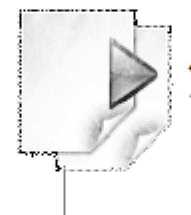


图 16-33　边框颜色属性的显示效果

16.5.15　边框宽度属性 border-width

边框宽度属性(border-width)用来统一定义边框的显示宽度。在边框宽度属性中，使用的属性值和相应的边框单侧宽度属性中相同。



注意

使用边框宽度属性，无法定义某一个方向上边框显示独立的宽度值。

下面是一个使用边框宽度属性的实例，其代码如下所示。

例程 16-31　border-width.html

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>无标题文档</title>
```

```
05    <style>
06    div{
      border-width:30px;
      border-style: double;
      width:300px;
      height:120px;
   }
07    </style>
08    </head>
09    <body>
      <div></div>
10    </body>
11    </html>
```

在以上的代码中，06 行定义边框样式为双线。代码运行后的显示效果如图 16-34 所示。

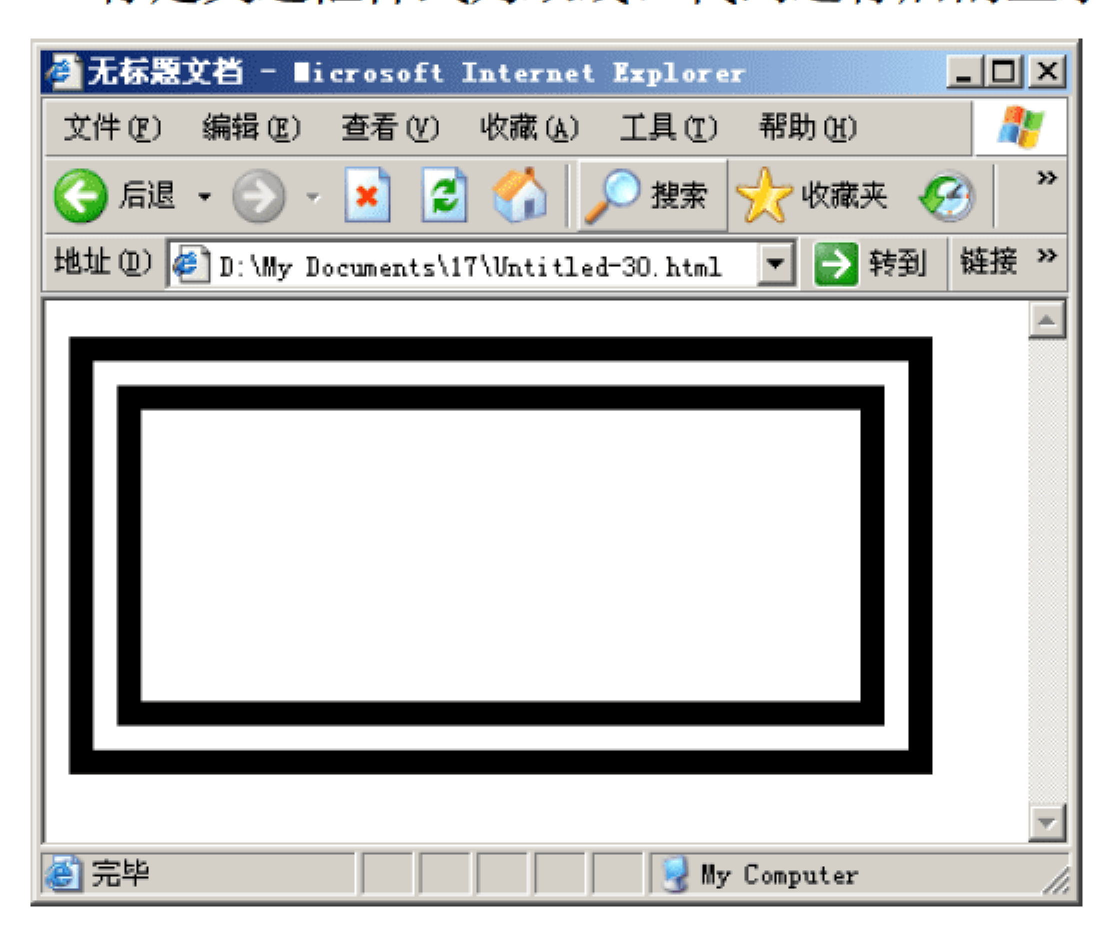


图 16-34　边框宽度属性的显示效果

16.5.16　边框顶部综合属性 border-top

边框顶部综合属性(border-top)用来统一定义边框顶部的显示效果。边框顶部综合属性是边框几个顶部属性的简化写法，其中使用的依然是边框样式、边框宽度、边框颜色等几个属性值，其语法结构如下所示。

```
border-top:border-top-style border-top-width border-top-color;
```

在边框顶部综合属性中，所有的背景属性值之间，使用空格分隔。当属性值的顺序发生变化时，并不影响边框属性的显示效果。

下面是一个使用边框顶部综合属性的实例，其代码如下所示。

例程 16-32　border-top.html

```
01    <html>
02    <head>
```

```
03  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
04  <title>css 属性实例</title>
05  <style>
06  div{
  border-top:20px #666666 double;
  width:500px;
  height:50px;
  }
07  </style>
08  </head>
09  <body>
    <div></div>
10  </body>
11  </html>
```

在以上的代码中，06 行中使用边框顶部综合属性，定义了顶部边框的宽度为 20px，颜色为黄色，边框样式为双线。同时定义了元素的宽度、高度，用来显示更好地显示元素的边框。以上代码的显示效果如图 16-35 所示。

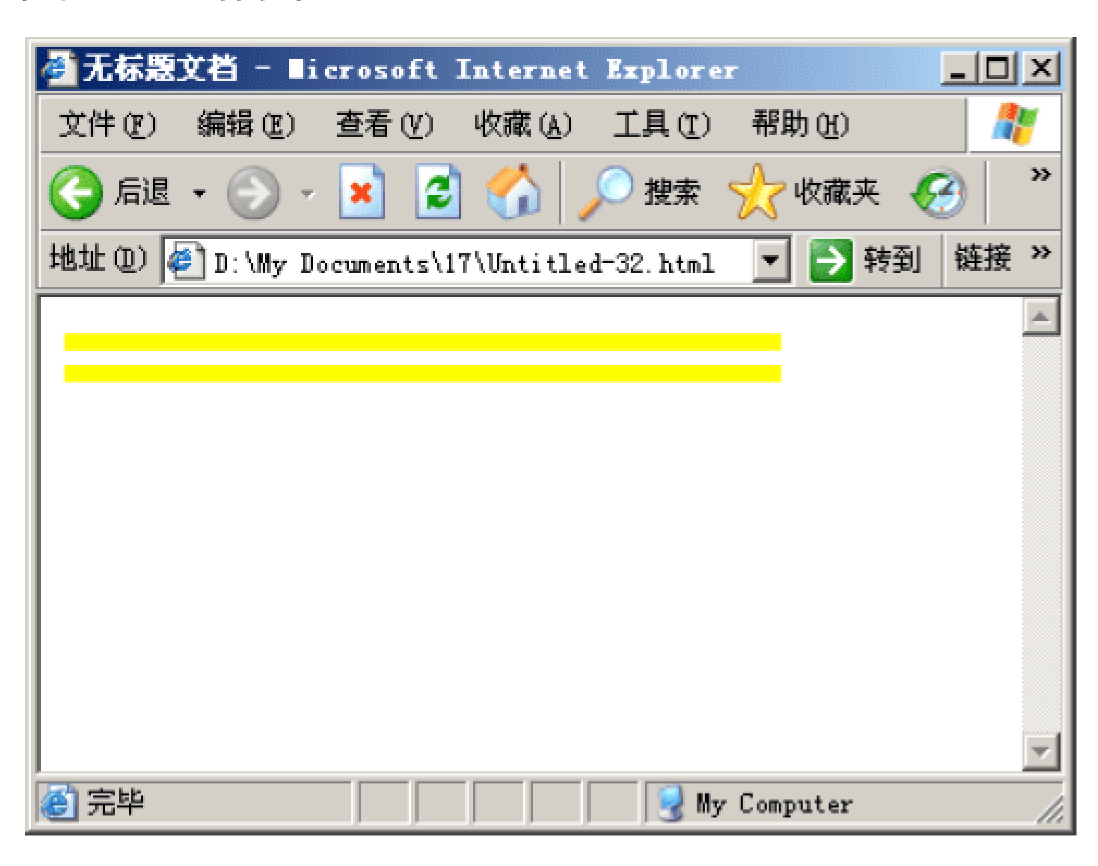


图 16-35　边框顶部综合属性的显示效果

16.5.17　边框右侧综合属性 border-right

边框右侧综合属性(border-right)用来统一定义边框右侧的显示效果。边框右侧综合属性是边框几个右侧属性的简化写法，其中使用的依然是边框样式、边框宽度、边框颜色等几个属性值，其语法结构如下所示。

```
border- right:border- right -style border- right -width border- right -color;
```

在边框右侧综合属性中，所有的背景属性值之间，使用空格分隔。当属性值的顺序发生变化时，并不影响边框属性的显示效果。使用方法和边框顶部综合属性一样，这里就不举实例叙述了。

16.5.18 边框底部综合属性 border-bottom

边框底部综合属性(border-bottom)用来统一定义边框底部的显示效果。边框底部综合属性是边框几个底部属性的简化写法，其中使用的依然是边框样式、边框宽度、边框颜色等几个属性值，其语法结构如下所示。

```
border-bottom: border-bottom-style border-bottom-width border-bottom-color;
```

在边框底部综合属性中，所有的背景属性值之间，使用空格分隔。当属性值的顺序发生变化时，并不影响边框属性的显示效果。使用方法和边框顶部综合属性一样，这里就不举实例叙述了。

16.5.19 边框左侧综合属性 border-left

边框左侧综合属性(border-left)用来统一定义边框左侧的显示效果。边框左侧综合属性是边框几个左侧属性的简化写法，其中使用的依然是边框样式、边框宽度、边框颜色等几个属性值，其语法结构如下所示。

```
border-left: border-left-style border-left-width border-left-color;
```

在边框左侧综合属性中，所有的背景属性值之间使用空格分隔。当属性值的顺序发生变化时，并不影响边框属性的显示效果。使用方法和边框顶部综合属性一样，这里就不举实例叙述了。

16.5.20 边框综合属性 border

边框综合属性(border)用来统一定义所有边框的显示效果。边框综合属性中，使用的依然是边框样式、边框宽度、边框颜色等几个属性值。其语法结构如下所示。

```
border:border-style border-width border-color;
```

在边框综合属性中，所有的背景属性值之间，使用空格分隔。当属性值的位置发生变化时，并不影响边框的显示效果。

下面是一个使用边框综合属性的实例，其代码如下所示。

例程 16-33 border.html

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>无标题文档</title>
<style>
div{
border:20px red double;
width:300px;
height:90px;
}
```

```
07  </style>
08  </head>
09  <body>
    <div></div>
10  </body>
11  </html>
```

在以上的代码中，06 行中使用边框综合属性，定义了边框的宽度为 20px，颜色为红色，边框样式为双线。同时定义了元素的宽度、高度，用来显示更好地显示元素的边框。以上代码的显示效果如图 16-36 所示。

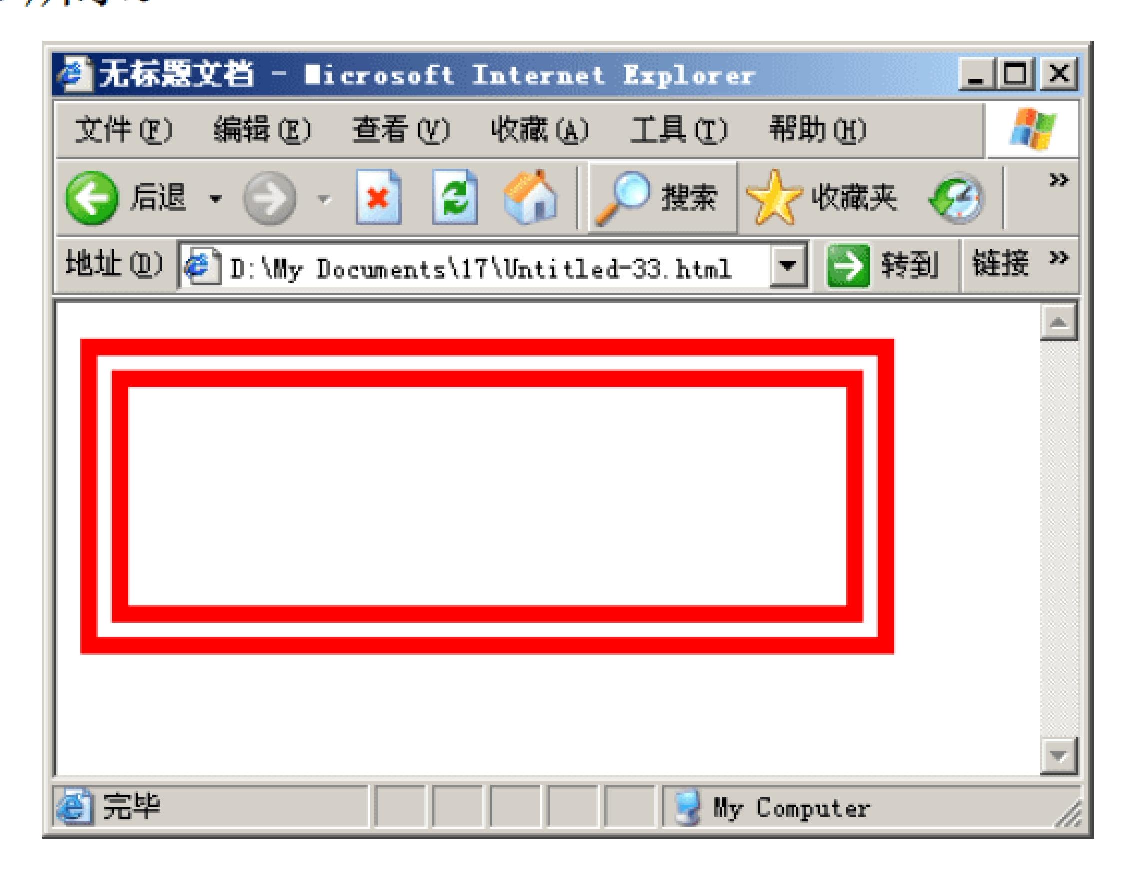


图 16-36　边框综合属性的显示效果

16.6 元素的边界

元素的边界，用来分隔元素和与其相邻的元素。在 CSS 中，边界属性分为 4 个单侧边界属性，包括顶部边界属性(margin-top)、右边界属性(margin-right)、下边界属性(margin-bottom)、左边界属性(margin-left)、以及边界综合属性(margin)等。

在 CSS 中，可以独立定义某个边界的属性值，也可以统一定义所有边界的属性值，下面分别进行讲解。

16.6.1　顶部边界属性 margin-top

顶部边界属性(margin-top)用来定义元素顶部边界的显示效果。在顶部边界属性中，可以使用长度值、百分比值和 auto 值。其语法结构如下所示。

```
margin-top: length | precent | auto;
```

注意

如果在单侧边界属性中使用百分比属性值，则最终的取值，取决于父元素中定义的元素宽度(关于父元素的具体内容，将在后面的 16.7 节中详细讲解)。

下面是一个使用顶部边界属性的实例，其代码如下所示。

例程 16-34 margin-top.html

```
01  <html>
02  <head>
03  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
04  <title>无标题文档</title>
05  <style>
06  .top
{
    border:2px red solid;
    width:300px;
    height:50px;
}
07  .bottom
{
    margin-top:50px;
    border:2px #000000 solid;
    width:300px;
    height:50px;
}
08  </style>
09  </head>
10  <body>
    <div class="top"></div>
    <div class="bottom"></div>
11  </body>
12  </html>
```

在以上的代码中，06 和 07 行中分别定义顶部边界属性值为 50px。同时定义了元素的宽度、高度，以及边框的宽度、颜色、样式等属性，其目的是用来显示元素边界属性的效果。以上代码的显示效果如图 16-37 所示。

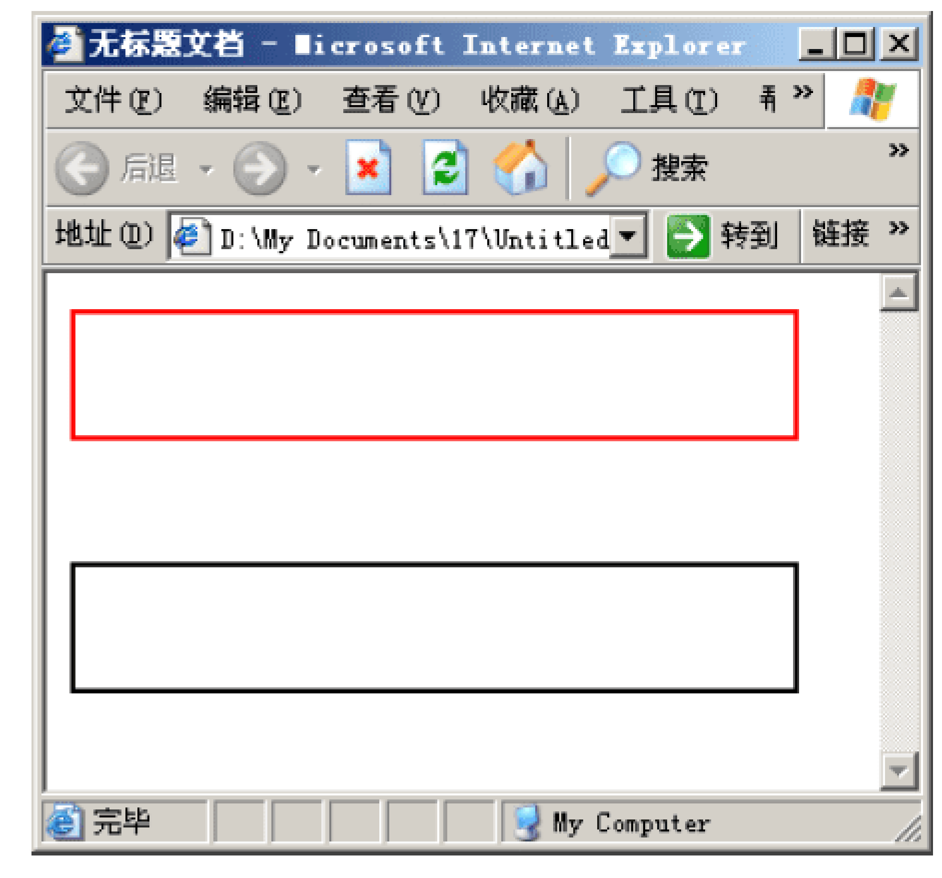


图 16-37 顶部边界属性的显示效果

16.6.2 右侧边界属性 margin-right

右侧边界属性(margin-right)用来定义元素右侧边界的显示效果。在右侧边界属性中，可以使用长度值、百分比值和 auto 值。其语法结构如下所示。

```
margin- right: length | precent | auto;
```

下面是一个使用右侧边界属性的实例，其代码如下所示。

例程 16-35 margin-right.html

```
01  <html>
02  <head>
03  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
04  <title>无标题文档</title>
05  <style>
06  .in
    {
    margin-right:80px;
    border:4px #000000 solid;
    background:yellow;
    width:300px;
    height:50px;
}
07  .main
{
    border:4px #000000 double;
}
08  </style>
09  </head>
10  <body>
    <div class="main">
    <div class="in"></div></div>
11  </body>
12  </html>
```

在以上的代码中，06 和 07 行使用了嵌套的<div>元素。在内部的<div>元素中，定义右侧边界属性值为 80px。同时定义了元素的宽度、高度，以及边框的宽度、颜色、样式等属性，其目的是用来显示元素边界属性的效果。以上代码的显示效果如图 16-38 所示。

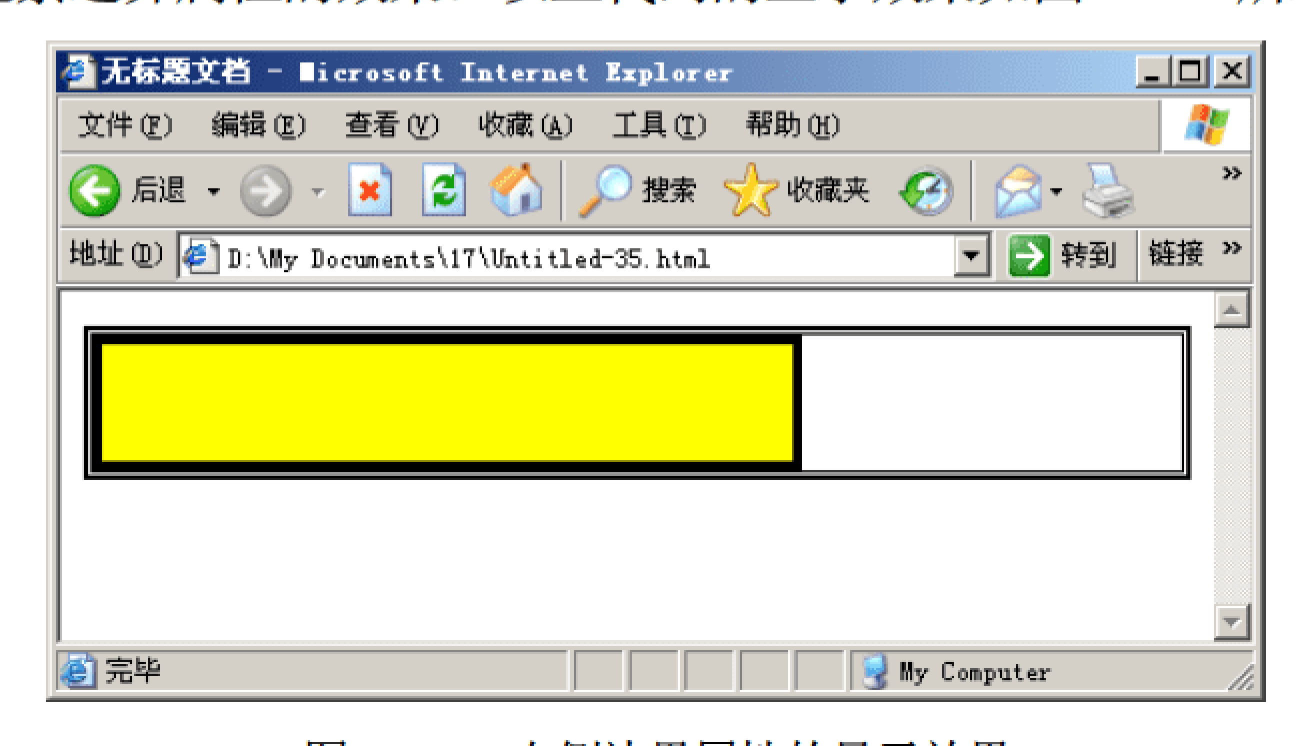


图 16-38 右侧边界属性的显示效果

16.6.3 底部边界属性 margin-bottom

底部边界属性(margin-bottom)用来定义元素底部边界的显示效果。在底部边界属性中，可以使用长度值、百分比值和 auto 值。其语法结构如下所示。

```
margin-bottom: length | precent | auto;
```

下面是一个使用底部边界属性的实例，其代码如下所示。

例程 16-36 margin-bottom.html

```
01  <html >
02  <head>
03  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
04  <title>无标题文档</title>
05  <style>
06  .in
  {
  margin-bottom:100px;
  border:4px #000000 solid;
  background:yellow;
  width:300px;
  height:80px;
  }
07  .main
  {
    height:80px;
    border:4px #000000 dotted;
  }
08  </style>
09  </head>
10  <body>
    <div class="main">
    <div class="in"></div></div>
11  </body>
12  </html>
```

在以上的代码中，使用了嵌套的<div>元素。06 行在内部的<div>元素中，定义底部边界属性值为 100px。同时定义了元素的宽度、高度，以及边框的宽度、颜色、样式等属性，其目的是用来显示元素边界属性的效果。以上代码的显示效果如图 16-39 所示。

注意

在 IE 6.0 中，一定要定义父元素的高度，否则子元素的底部边界属性值将无法显示(关于 0 边界的问题，在 16.7 节中将详细讲解)。

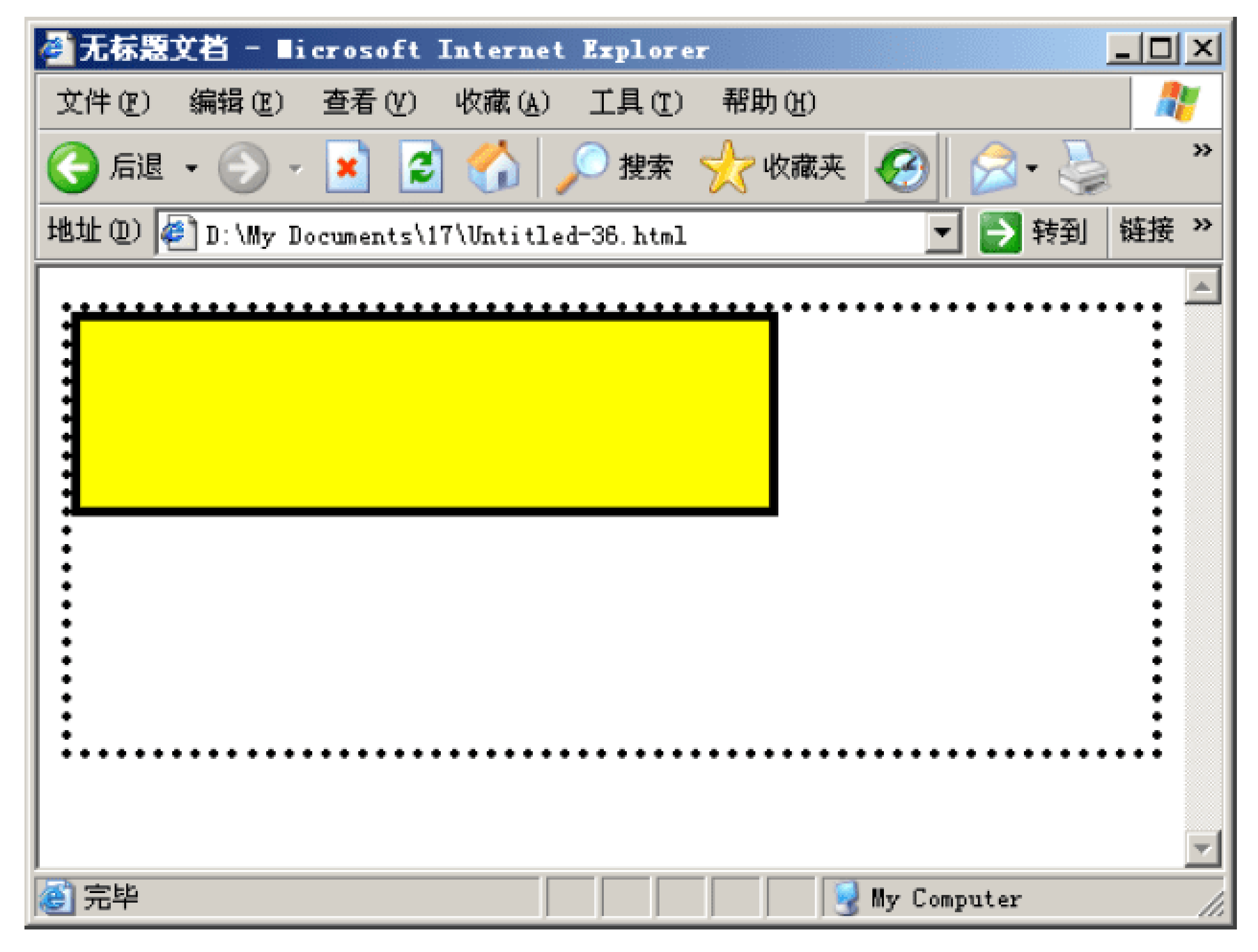


图 16-39　底部边界属性的显示效果

16.6.4　左侧边界属性 margin-left

左侧边界属性(margin-left)用来定义元素左侧边界的显示效果。在左侧边界属性中，可以使用长度值、百分比值和 auto 值。其语法结构如下所示。

```
margin-left: length | precent | auto;
```

下面是一个使用左侧边界属性的实例，其代码如下所示。

例程 16-37　margin-left.html

```
<html >
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>无标题文档</title>
<style>
div
 {
   margin-left:100px;
   border:2px #000000 solid;
   background:yellow;
   width:300px;
   height:150px;
 }
.main
 {
   height:80px;
   border:4px #000000 dotted;
 }
</style>
</head>
<body>
```

```
    <div class="main">
    <div></div></div>
11  </body>
12  </html>
```

在以上的代码中，06 行在<div>元素中，定义左侧边界属性值为 100px。同时定义了元素的宽度、高度，以及边框的宽度、颜色、样式等属性，其目的是用来显示元素边界属性的效果。以上代码的显示效果如图 16-40 所示。

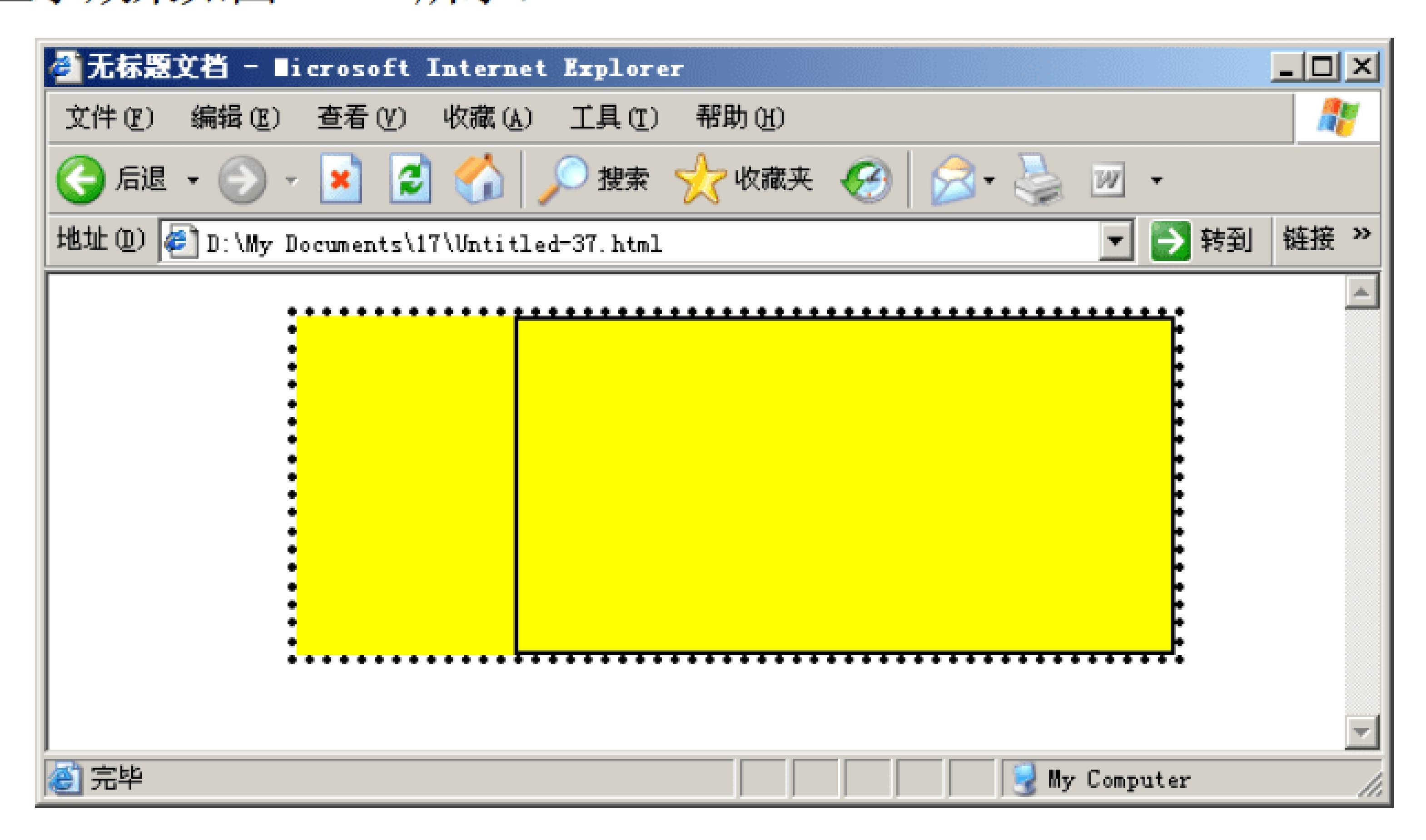


图 16-40　左侧边界属性的显示效果

16.6.5　边界综合属性 margin

边界综合属性(margin)用来统一定义边界的显示效果。在边界综合属性中，可以使用长度值、百分比值和 auto 值。其语法结构如下所示。

```
margin: length | precent | auto;
```

注意

如果在边界综合属性中使用百分比属性值，则最终的取值取决于父元素中定义的元素宽度(关于父元素的具体内容，将在 16.7 节中详细讲解)。

下面是一个使用边界综合属性的实例，其代码如下所示。

例程 16-38　example.html

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>无标题文档</title>
<style>
.include{
```

```
        margin:60px;
        border:2px red solid;
        width:300px;
        height:50px;
        }
07    .main{
        border:2px #000000 dotted;
        width:80px;}
08    </style>
09    </head>
10    <body>
        <div class="main"><div class="include"></div></div>
11    </body>
12    </html>
```

在以上的代码中，使用了嵌套的<div>元素。06 行在内部的<div>元素中，定义边界综合属性值为 60px。同时定义了元素的宽度、高度，以及边框的宽度、颜色、样式等属性，其目的是用来显示元素边界属性的效果。以上代码的显示效果如图 16-41 所示。

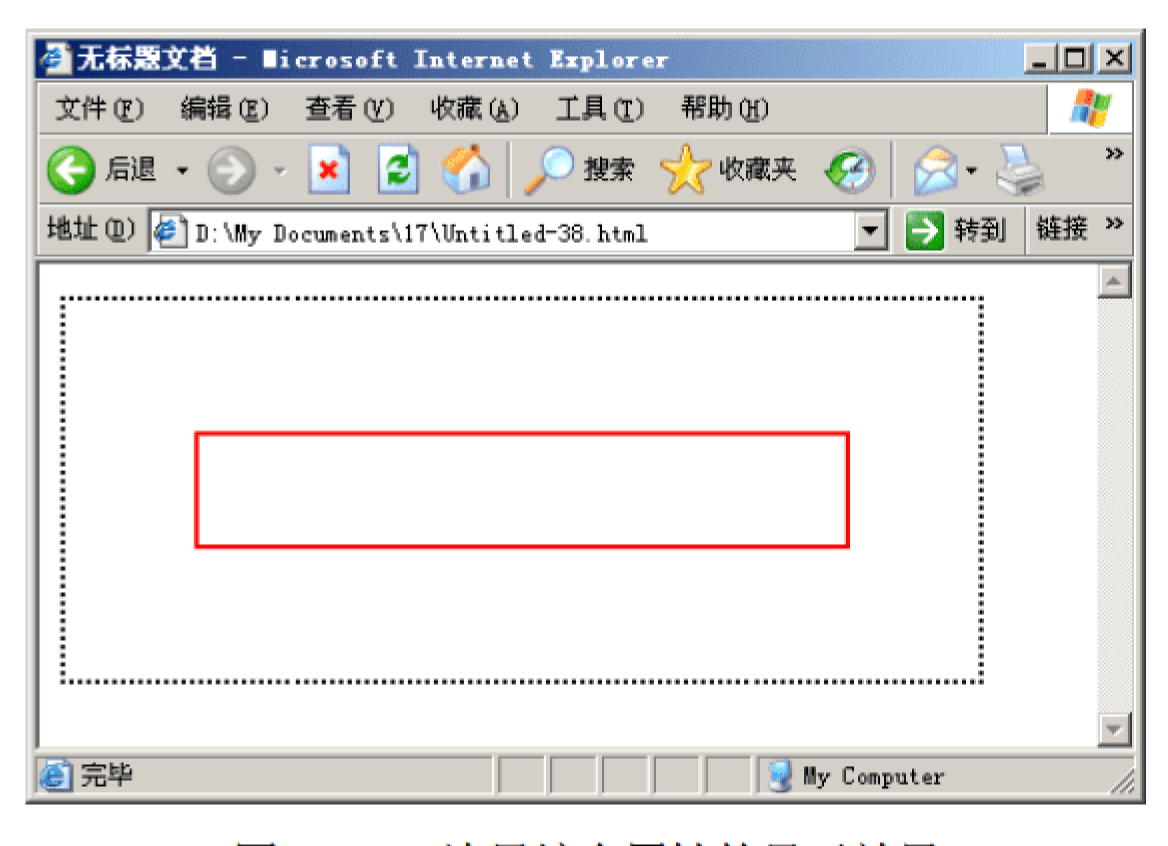


图 16-41　边界综合属性的显示效果

16.6.6　边界与背景

在元素的边界部分，不会显示元素本身定义的背景颜色或者背景图片，但是可以显示父元素中定义的背景颜色或背景图片。

下面是一个关于边界与背景属性的实例，其代码如下所示。

例程 16-39　example.html

```
<html >
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>无标题文档</title>
<style>
.include{
    margin:60px;
```

```
        background:red;
        width:300px;
        height:60px;
        }
07      .main{
        border:2px #000000 dotted;
        background:yellow;
        width:60px;
        }
08      </style>
09      </head>
10      <body>
        <div class="main"><div class="include"></div></div>
11      </body>
12      </html>
```

以上的代码中，06 行在子元素<div>标签中定义边界属性值为 60px，背景颜色为红色。在父元素中定义背景颜色为黄色。此时在自元素的边界区域会显示父元素的背景，其页面的显示效果如图 16-42 所示。

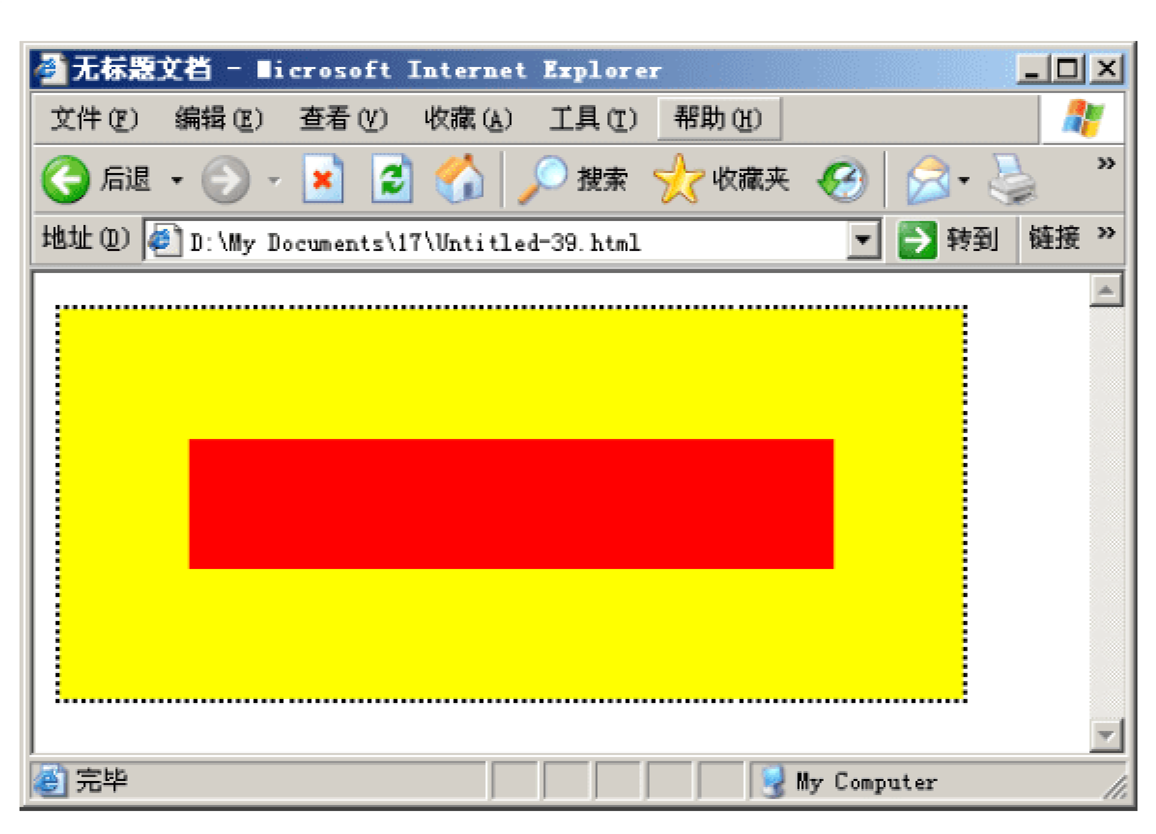
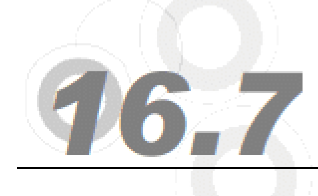


图 16-42　边界区域显示父元素背景的效果



16.7 嵌套元素的大小和距离

在嵌套的元素中，由于父元素和子元素中都可以使用边界和补白属性，所以元素属性之间会互相影响。在不同的浏览器中，对这种影响也有不同的解释。本章将讲解在 IE 6.0 版本中，嵌套元素之间的关系。

16.7.1　父元素和子元素

在使用元素制作页面的时候，通常会遇到某一个元素中包含另一个元素的情况。通常称被包含的元素为子元素，包含子元素的元素为父元素。

在默认的情况下，在子元素中定义的相关属性，会影响父元素的显示效果。同时在父元素中定义的属性，也可以制约子元素的显示效果。下面是包含子元素和父元素的实例，其代码如下所示。

例程 16-40 example.html

```
01 <html>
02 <head>
03 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
04 <title>无标题文档</title>
05 <style>
06 .main{
   background:yellow;
   width:50px;}
07 .include{
   width:300px;
   height:150px;
   }
08 </style>
09 </head>
10 <body>
   <div class="main"><div class="include"></div></div>
11 </body>
12 </html>
```

以上的代码中，06 行定义父元素的背景颜色为黄色，宽度为 50px。07 行子元素的宽度为 300px，高度为 150px。此时，由于子元素的宽度大于父元素的宽度，所以最终父元素的宽度会和子元素相同。由于父元素中未定义高度，父元素的高度将显示子元素的高度。由于子元素中未定义背景，所以显示父元素中的背景。以上代码的显示效果如图 16-43 所示。

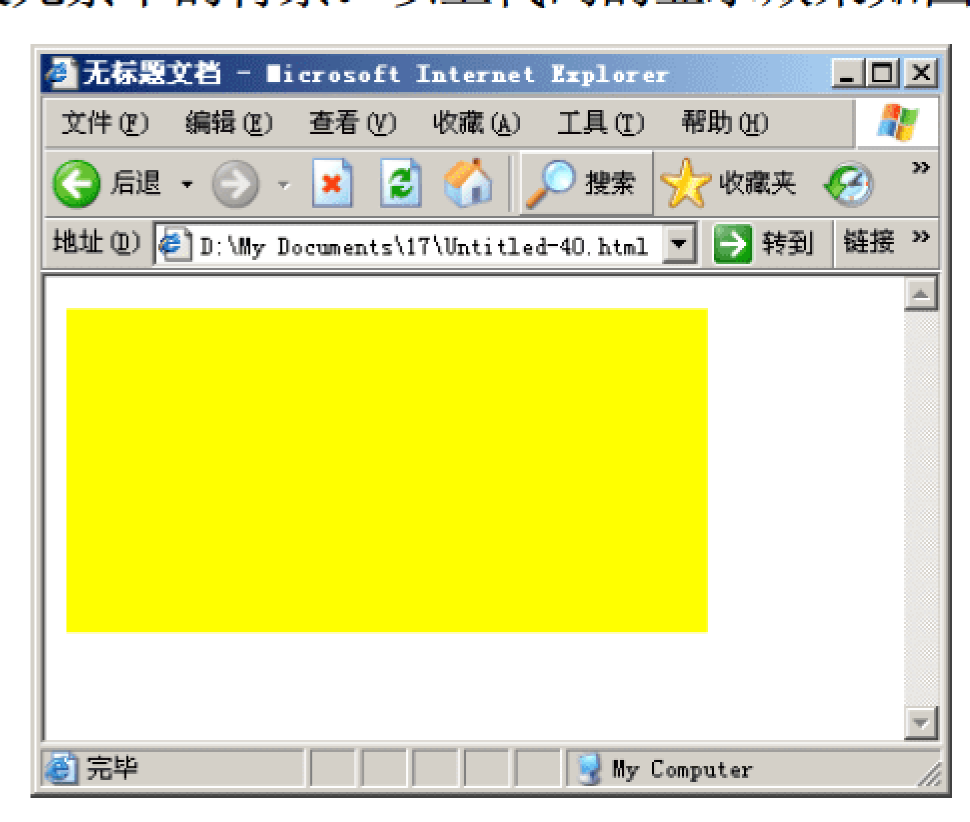


图 16-43 嵌套元素的显示效果

16.7.2 子元素中使用边界属性，父元素未定义大小

当子元素中使用边界属性，同时父元素未定义大小的时，在 IE 6.0 中，父元素和子元素之

间在垂直方向上，边界属性将无法正常显示。

下面是子元素中使用边界属性，而父元素未定义大小的实例，其代码如下所示。

例程 16-41 example.html

```
01  <html >
02  <head>
03  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
04  <title>无标题文档</title>
05  <style>
06  .main{
    border:4px double red;}
07  .include{
    margin:60px;
    border:2px solid yellow;
    width:300px;
    height:60px;
    }
08  </style>
09  </head>
10  <body>
    <div class="main"><div class="include"></div></div>
11  </body>
12  </html>
```

以上的代码中，07 行在子元素中定义了边界属性为 60px，同时定义了子元素的大小为 300px×60px，边框属性值为黄色实线边框，边框宽度为 2px。06 行在父元素中定义了边框属性值为红色双线边框，边框宽度为 4px。

由于 IE 6.0 浏览器对 CSS 的解释不同，所以在 IE 6.0 中，子元素与父元素在垂直方向上无法显示定义的边界属性。以上代码的显示效果如图 16-44 所示。

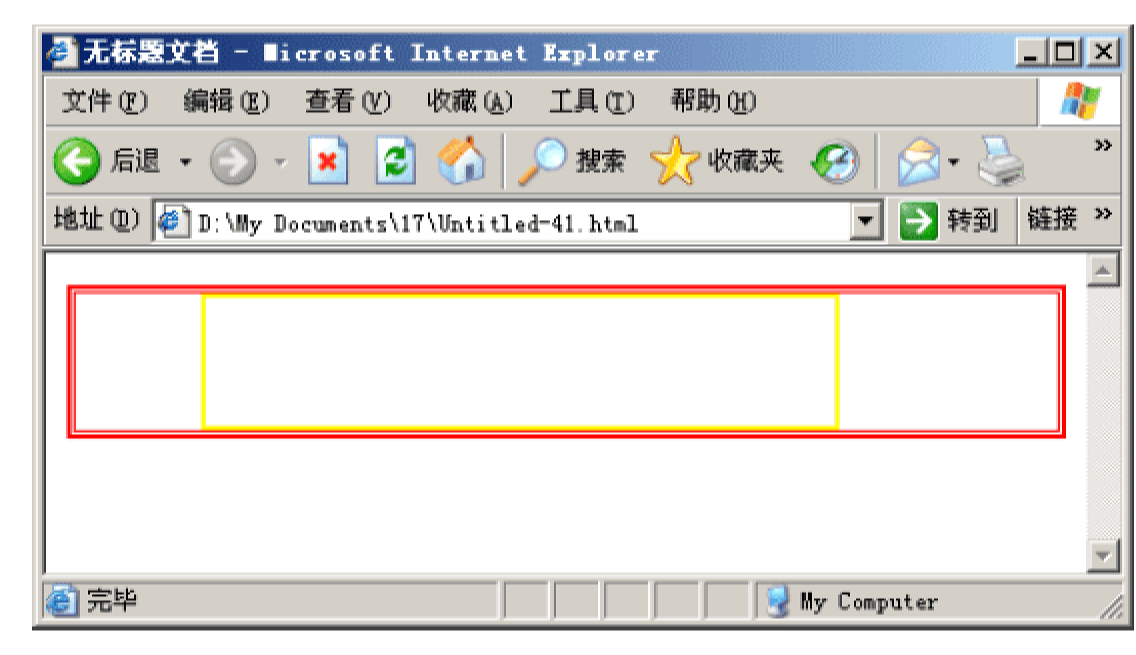


图 16-44 IE6.0 中实例的显示效果

16.7.3 子元素中使用边界属性，父元素中使用补白属性

当子元素中使用边界属性，同时父元素中使用补白属性的时候，在 IE 浏览器中，元素顶部会将子元素的边界属性和父元素的补白属性值重叠，底部会将子元素的边界属性和父元素的补白属性值相加。

下面是子元素中使用边界属性，父元素中使用补白属性的实例，其代码如下所示。

例程 16-42 example.html

```
<html>
<head>
```

```
03  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
04  <title>无标题文档</title>
05  <style>
06  .main{
    padding:60px;
    border:2px solid red;
    width:100px;
    height:50px;
    }
07  .include{
    margin:60px;
    border:2px double yellow;
    width:300px;
    height:50px;
    }
08  </style>
09  </head>
10  <body>
    <div class="main"><div class="include"></div></div>
11  </body>
12  </html>
```

以上的代码中，07 行在子元素中定义了边界属性为 60px。06 行父元素中定义补白属性为 60px。由于 IE 6.0 浏览器对 CSS 的解释不同，所以在 IE 6.0 中，以上代码的显示效果如图 16-45 所示。

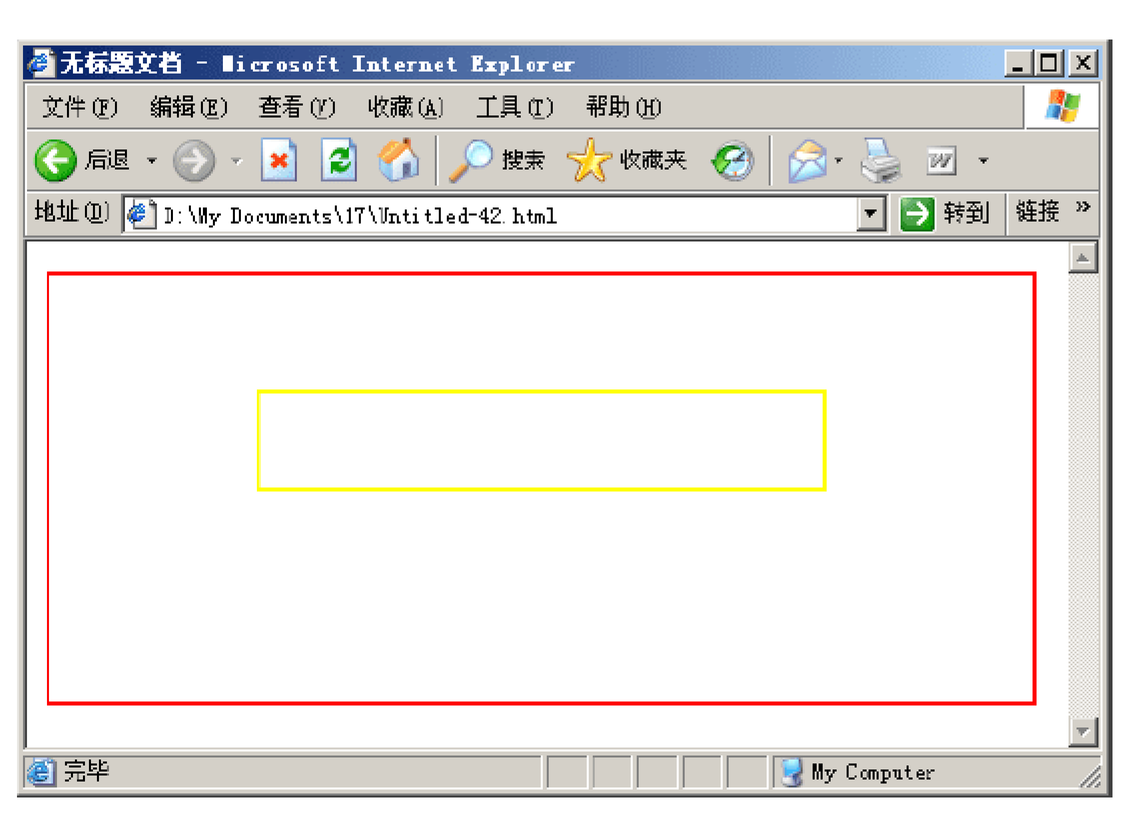


图 16-45　IE 6.0 中实例的显示效果

16.8 本章习题

一、选择题

1．以下关于内容与宽度、高度属性的关系说法中，错误的是：(　　)

A. 高度和宽度固定后，不论内容多少都不会改变他们的大小。

B. 高度和宽度固定后，内容过多时会改变他们的大小。

C. 语句 {height:80px;}这里定义页面的高度为 80px。内容超过这个高度时，高度会增加。

D. 语句 { width:200px;}这里定义页面的高度为 200px。内容超过这个长度会换行。

2．用于设置元素内容显示宽度的属性是(　　)。

A. Size　　B　MaxLength　C. Value　D. width

3．表示内容和边框之间的距离的属性是(　　)。

A.< margin>　B. <padding>　C. < border>　D. <object>

4．若要在网页中内容和边框之间的上下之间距离为 20px，左右之间的距离为 30px，以下代码中，正确的是(　　)。

A. < margin 20px 0 20px 0;>

B. <margin 20px 30px;>

C. <padding 20px 0 20px 0;>

D. <padding 20px 30px;>

5．使元素背景图片随浏览器滚条的拖动而滚动的属性是(　　)。

A. scroll　　B. fixed　　C. center　　D. no-repeat.

6．使元素外边距上下为 5px，左右为 10px，以下代码中，正确的是(　　)。

A. < margin 5px 0 10px 0;>

B. < margin 5px 10px;>

C. < padding 5px 0 10px 0;>

D. < padding 5px 10px;>

二、填空题

1. <background:url(images/round.jpg) repeat-x center #cccccc;>这段代码表示的意思是________。

2. <padding:20px 30px 40px 50px;>表示__________，<padding:20px 30px 40px;>表示__________。

3. <border:12px #ffff03 double;>表示____________________

4. < margin:60px;>表示________________。

三、实战练习

1．制作一个页面，要求边框宽度为 4px，外边距为左右 20px，上下 5px，补白为 50px。

2．制作一个页面，要求背景图片随浏览器滚条的拖动而滚动。

Chapter 17

CSS控制元素的定位

在 CSS 中，可以通过定位属性控制元素的显示位置。使用定位属性可以设置元素的重叠、相对位置等，也可以制作出各种特殊的显示效果。在 CSS 中，可以使用绝对定位属性值和相对定位属性值对内容进行合理的安排。

本章主要内容有:

◎ 掌握元素定位的方法。

◎ 重点理解绝对定位和相对定位。

◎ 学会合理地重叠定位元素。



17.1 元素的定位

在网页中，如果元素中未定义任何定位和布局属性，元素会按照各自默认的方式排列。块元素会按照换行的方式显示，内联元素会按照同行的方式显示。通过使用定义属性，可以将元素定义到任何需要显示的位置。

17.1.1 元素的定位属性 position

元素的定位属性(position)用来定义元素的定位方式。在定位属性中，可以使用 4 种属性值，分别为 static、absolute、fixed 和 relative，其语法结构如下所示。

```
position : static | absolute | fixed | relative;
```

其中每个属性值的含义，如下所示。

- static：默认值，元素按照自身默认的方式显示。
- absolute：绝对定位，以有含有定位属性的父元素为基准，按照边偏移属性中定义的属性值显示。
- fixed：浏览器不支持该属性。
- relative：相对定位，元素以自身位置为基准，按照边偏移属性中定义的属性值显示。

下面是一个使用元素的定位属性的实例，其代码如下所示。

例程 17-1 position.html

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title 元素的定位属性实例</title>
<style>
div{
position: relative;
left:200px;
top:50px;
border:3px solid #333333;
width:400px;
height:50px;
}
</style>
</head>
<body>
<div>这是一个使用定位属性 position 中 relative 的实例，在使用定位属性的时候，还要使用其他的辅助属性，才能正常显示。</div>
</body>
</html>
```

以上的代码中，06 行定义了元素的定位属性值为相对定位，使用边偏移属性确定元素左侧偏移值为 200px，顶部偏移值均为 50px。此时元素将以自身位置为基准，按照边偏移属性中定义的属性值显示，其显示效果如图 17-1 所示。

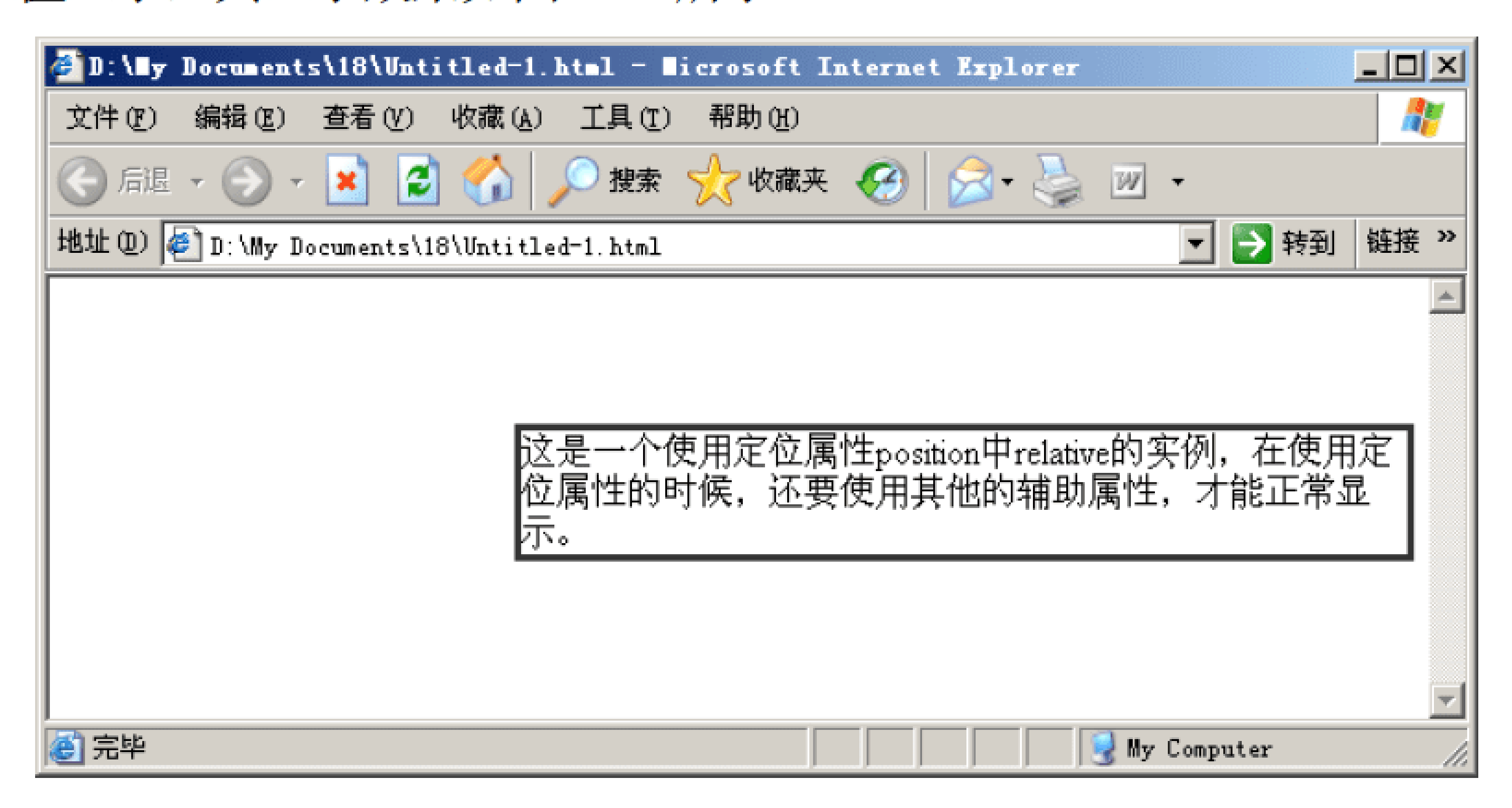


图 17-1　定位属性的显示效果

17.1.2　上边偏移属性 top

元素的上边偏移属性(top)用来定义元素顶部偏移位置的大小。在上边偏移属性中，可以使用的属性值为默认、长度值、百分比，其语法结构如下所示。

```
top:auto | length | percent;
```



注意

使用百分比值的时候，只有在定义元素定位属性值为绝对定位(absolute)或者相对定位(relative)的时候，才能正常显示。

下面是一个使用上边偏移属性的实例，其代码如下所示。

例程 17-2　top.html

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>无标题文档</title>
<style>
div{
top:20%;
position:absolute;
border:4px solid #000000;
background:#FFFF00;
width:400px;
```

```
        height:200px;
        }
07      </style>
08      </head>
09      <body>
        <div>这是使用顶部边偏移属性的实例。</div>
10      </body>
11      </html>
```

以上的代码中，06 行定义<div>元素的顶部的偏移值为 20%，同时定义了元素的定位属性值为绝对定位。此时，元素垂直方向上会以<body>元素的顶部为基准显示。以上代码的显示效果，偏移的宽度大概是整个页面宽度的 20%。如图 17-2 所示。

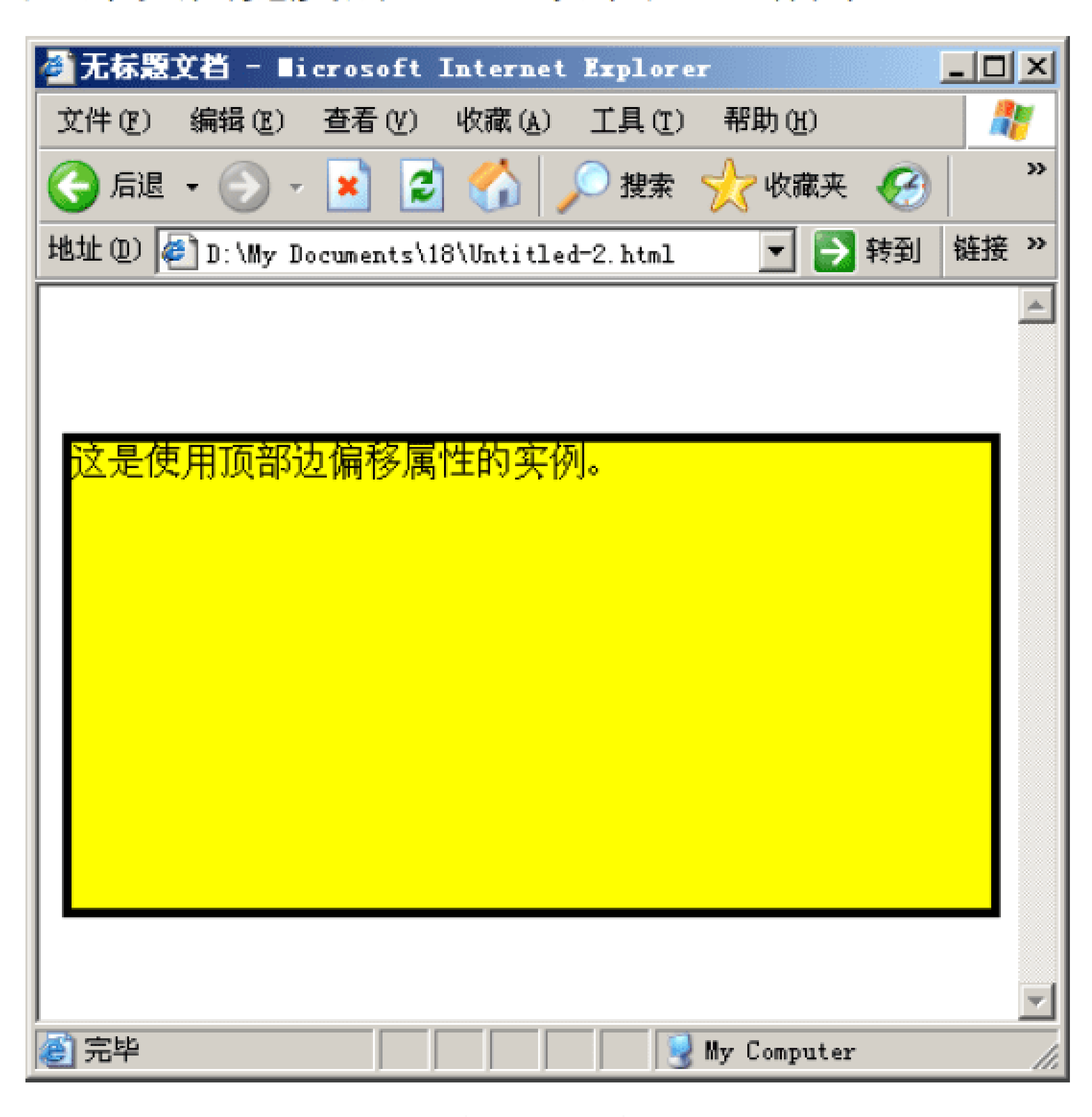


图 17-2　上边偏移属性的显示效果

17.1.3　右边偏移属性 right

元素的右边偏移属性(right)用来定义元素右侧偏移位置的大小。在右边偏移属性中，可以使用的也是为默认、长度值、百分比，其语法结构如下所示。

```
right:auto | length | percent;
```

在使用右边偏移属性的时候，如果未定义左边偏移属性，则元素的显示位置将根据浏览器窗口的大小改变而改变。

下面是一个使用右边偏移属性的实例，其代码如下所示。

例程 17-3　right.html

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
```

```
04  <title>无标题文档</title>
05  <style>
06  div{
    right:100px;
    position:absolute;
    border:4px solid #00000;
    background:#666666;
    width:200px;
    height:100px;
    }
07  </style>
08  </head>
09  <body>
    <div>这是使用右边偏移属性的实例。</div>
10  </body>
11  </html>
```

以上的代码中，06 行定义<div>元素的右边偏移值为 100px，同时定义了元素的定位属性值为绝对定位。此时，元素在水平方向上会以<body>元素的右侧为基准显示。以上代码运行后，在不同大小的浏览器窗口中的显示效果，如图 17-3 和图 17-4 所示。

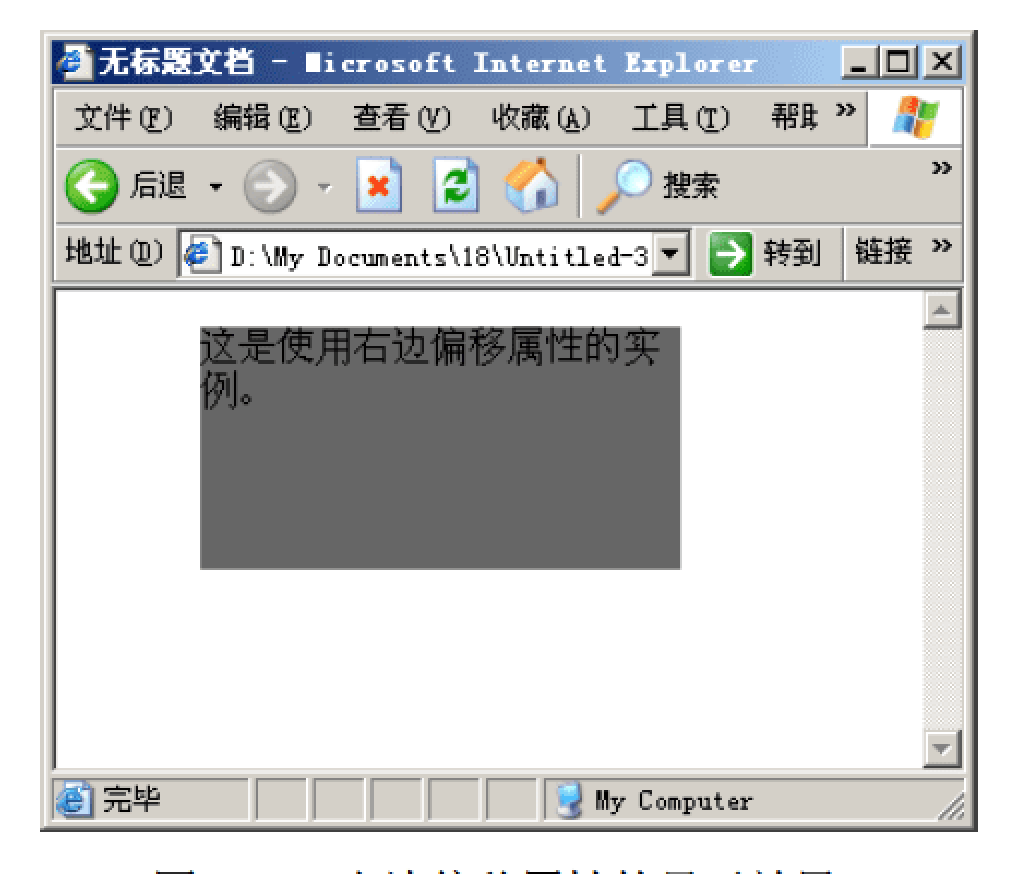


图 17-3　右边偏移属性的显示效果 1

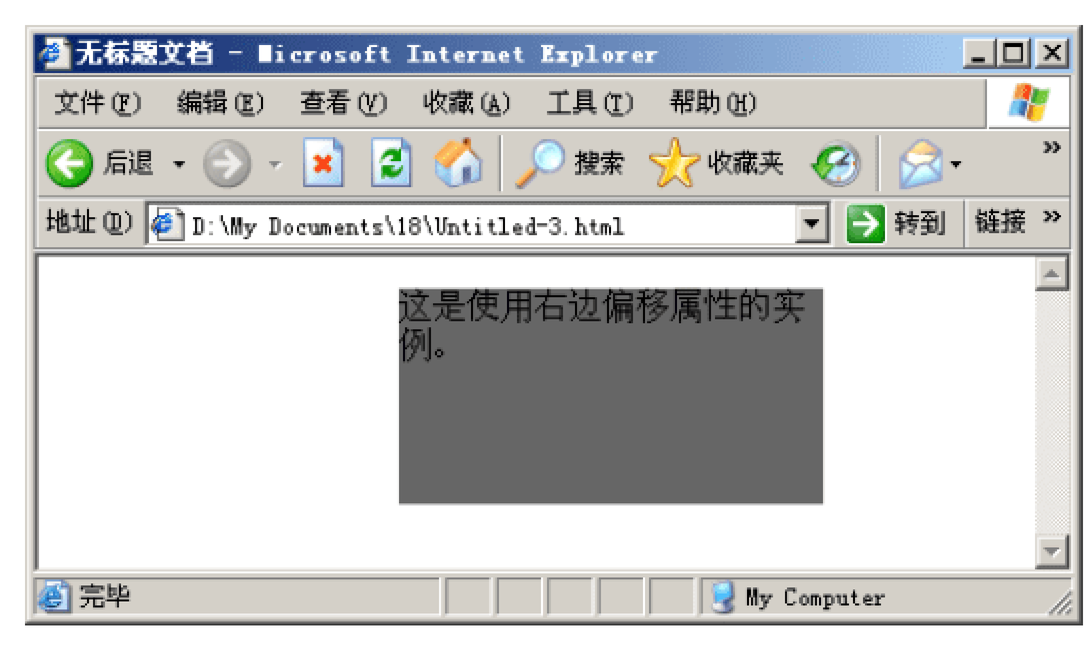


图 17-4　右边偏移属性的显示效果 2

17.1.4　下边偏移属性 bottom

元素的下边偏移属性(bottom)用来定义元素底部偏移位置的大小。在下边偏移属性中，可以使用的属性值也有默认、长度值、百分比，其语法结构如下所示。

```
top:auto | length | percent;
```

在使用下边偏移属性的时候，如果未定义上边偏移属性，则元素的显示位置将根据浏览器窗口的大小改变而改变。

下面是一个使用下边偏移属性的实例，其代码如下所示。

例程 17-4 bottom.html

```
01 <html xmlns="http://www.w3.org/1999/xhtml">
02 <head>
03 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
04 <title>无标题文档</title>
05 <style>
06 div{
   bottom:50px;
   position:absolute;
   border:3px solid #000000;
   background:#FF33CC;
   width:200px;
   height:60px;
   }
07 </style>
08 </head>
09 <body>
   <div>这是使用下边偏移属性的实例。</div>
10 </body>
11 </html>
```

以上的代码中，06 行定义<div>元素的下边偏移值为 50px，同时定义了元素的定位属性值为绝对定位。此时，元素在垂直方向上会以<body>元素的底部为基准显示。以上代码运行后，在不同大小的浏览器窗口中的显示效果，如图 17-5 和图 17-6 所示。

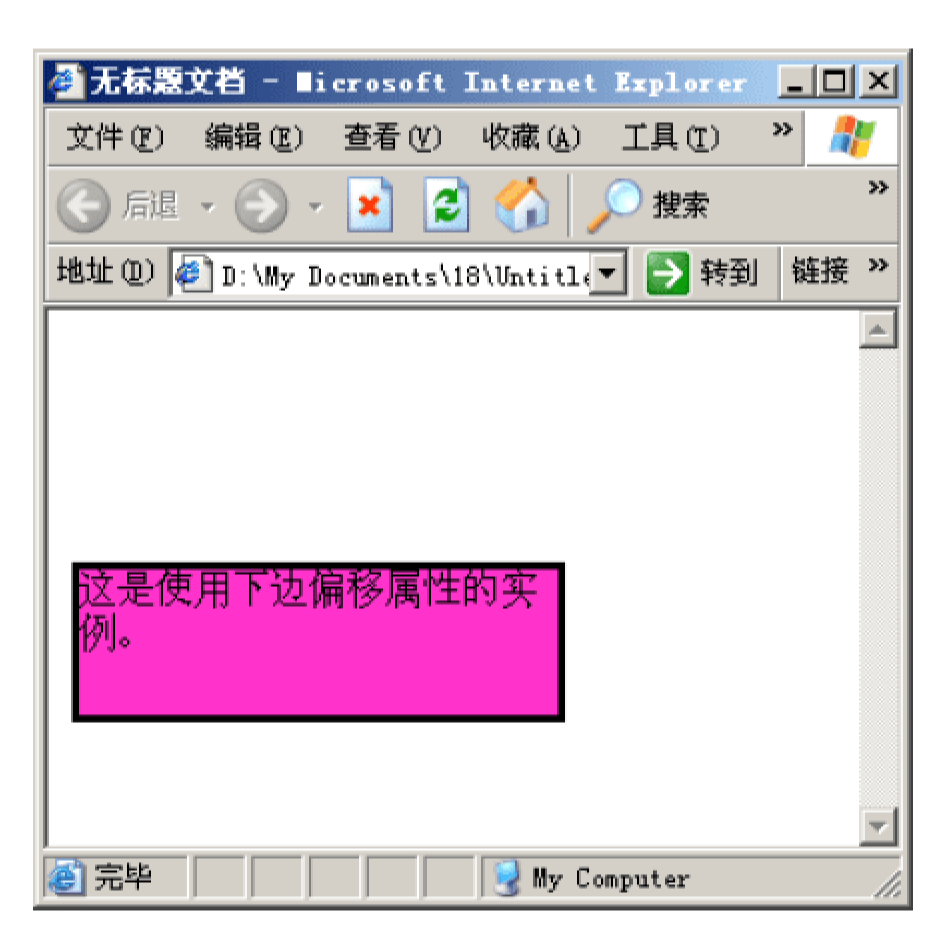


图 17-5 下边偏移属性的显示效果 1

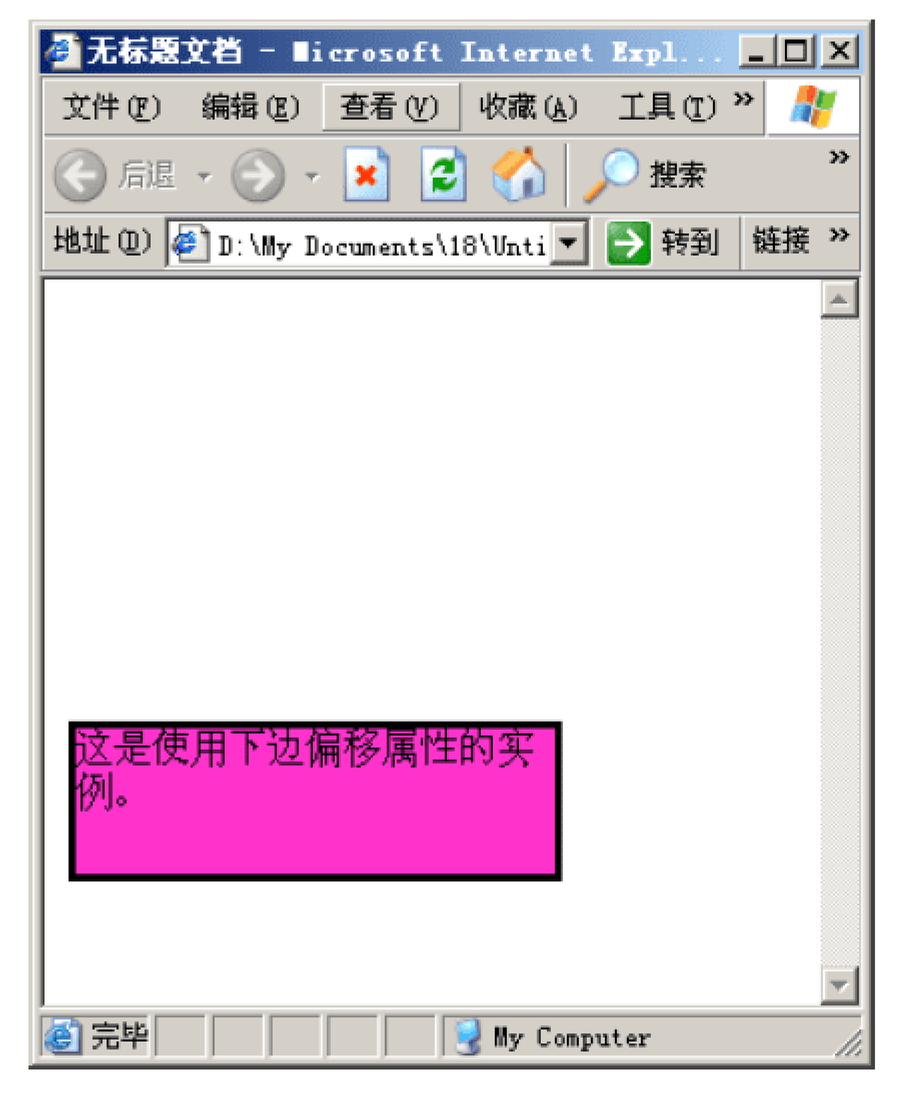


图 17-6 下边偏移属性的显示效果 2

17.1.5 左边偏移属性 left

元素的左边偏移属性(left)用来定义元素左边偏移位置的大小。在左边偏移属性中，可以使用的属性值也有默认、长度值、百分比，其语法结构如下所示。

```
left:auto | length | percent;
```

下面是一个使用左边偏移属性的实例，其代码如下所示。

例程 17-5 left.html

```
01 <html xmlns="http://www.w3.org/1999/xhtml">
02 <head>
03 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
04 <title>无标题文档</title>
05 <style>
06 div{
   left:200px;
   position:absolute;
   border:2px solid #000000;
   background:red;
   width:200px;
   height:50px;
   }
07 </style>
08 </head>
09 <body>
   <div>这是使用左边偏移属性的实例。</div>
10 </body>
11 </html>
```

以上的代码中，06 行定义<div>元素左边偏移值为 200px，同时定义了元素的定位属性值为绝对定位。此时，元素在水平方向上会以<body>元素的左侧为起点显示。以上代码的显示效果，如图 17-7 所示。

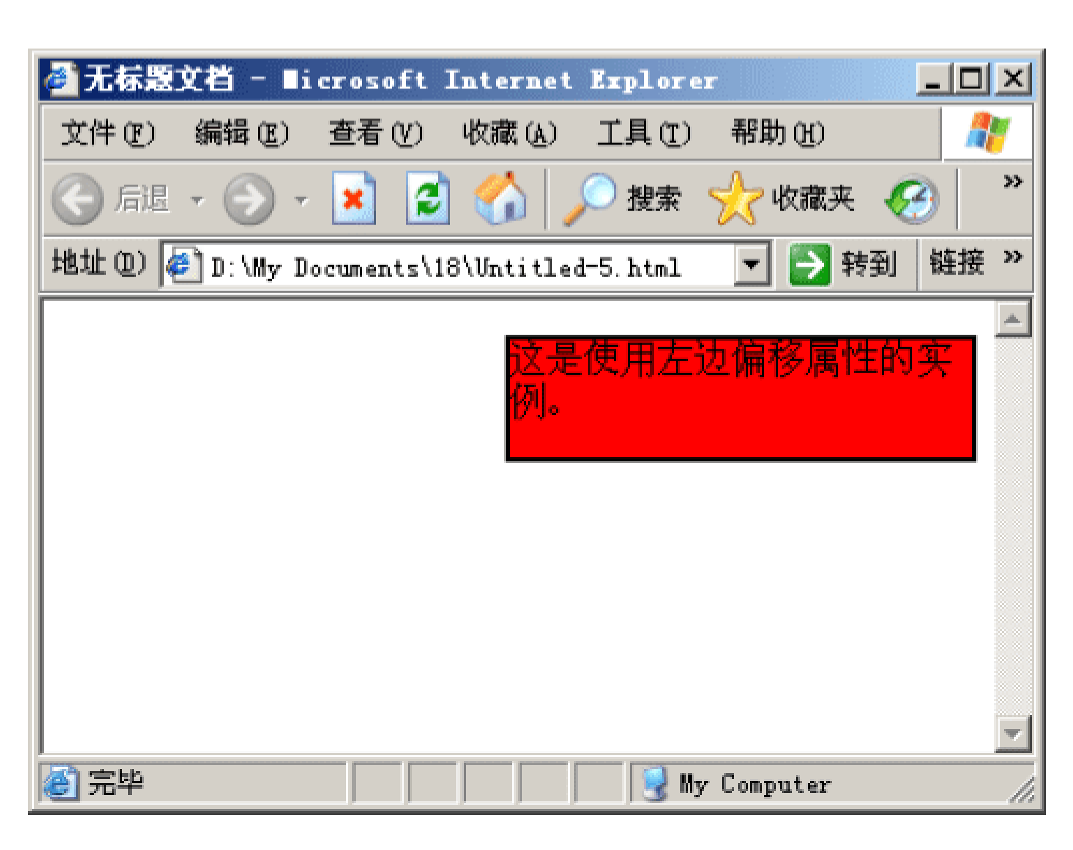


图 17-7 左边偏移属性的显示效果

17.2 绝对定位

绝对定位，用来确定元素相对定义父元素的绝对位置。在页面中，使用绝对定位的元素，

会从文档中独立地显示出来，所以使用绝对定位的元素可能会遮盖其他的页面元素。下面分别进行讲解。

17.2.1　绝对定位与父元素

在使用绝对定位的时候，定位的参照元素，是包含定位属性的父元素。如果没有这样的父元素，则元素按照<body>元素的位置确定显示位置。

下面是一个使用嵌套绝对定位元素的实例，其代码如下所示。

例程 17-6　example.html

```
01  <html xmlns="http://www.w3.org/1999/xhtml">
02  <head>
03  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
04  <title>无标题文档</title>
05  <style>
06  .main{
      position:absolute;
    top:50px;
    left:50px;
    width:300px;
    height:150px;
    border:2px solid #000000;}
07  .include{
    position:absolute;
    left:50%;
    top:50px;
    border:2px solid #333333;
    background:red;
    width:200px;
    height:50px;
  }
08  </style>
09  </head>
10  <body>
    <div class="main">
    <div class="include">这是嵌套绝对定位元素的实例。</div></div>
11  </body>
12  </html>
```

在以上的代码中，定义两个绝对定位的元素。06 行在父元素中使用绝对定位属性和边偏移属性，定义元素相对<body>元素的顶部显示位置和左侧显示位置均为 50px。07 行在子元素中，定义元素的左侧显示位置为 50%，顶部显示位置为 50px。以上代码的显示效果，如图 17-8 所示。

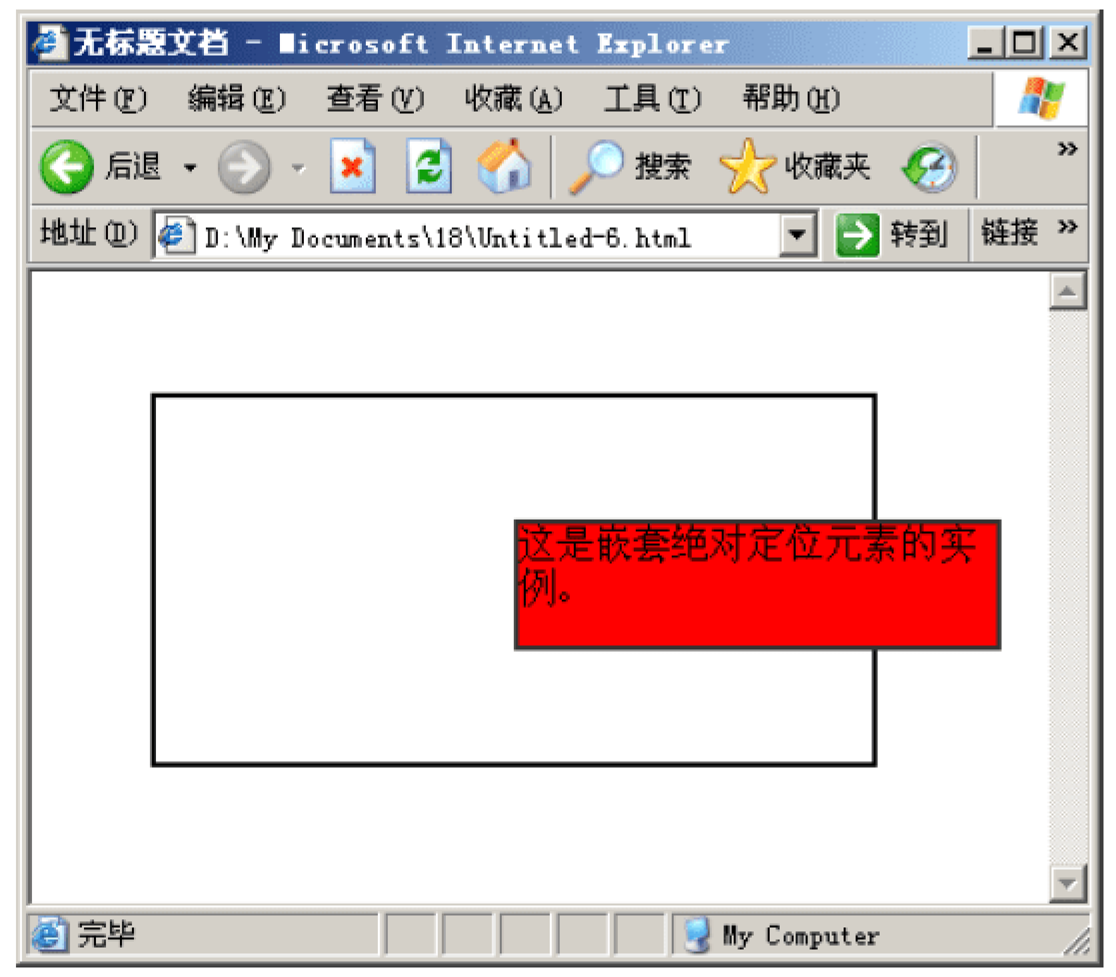


图 17-8 嵌套绝对定位元素的显示效果

从图 17-8 可以看出，当父元素中使用了绝对定位属性值的时候，子元素的位置会按照父元素的位置显示。同时子元素可能会覆盖父元素。

17.2.2 绝对定位与相邻元素

在使用绝对定位的时候，绝对定位元素会独立显示，并不影响其他元素的显示位置(但是有可能覆盖其他元素)。与绝对定位元素相邻的元素，会忽略绝对定位元素的存在，按照各自的显示方式正常显示。

下面是一个绝对定位与相邻元素的实例，其代码如下所示。

例程 17-7 example.html

```
01 <html xmlns="http://www.w3.org/1999/xhtml">
02 <head>
03 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
04 <title>无标题文档</title>
05 <style>
06 .div1{
   background:#FF0000;
   width:220px;
   height:110px;
   }
07 .absolute{
   position:absolute;
   top:50px;
   left:50px;
   width:320px;
   height:110px;
   background:#00CC33;}
08 .div2{
   background:blue;
```

```
           width:220px;
           height:110px;
       }
09     </style>
10     </head>
11     <body>
           <div class="div1"></div>
           <div class="absolute">这是绝对定位的元素。</div>
           <div class="div2"></div>
12     </body>
13     </html>
```

在以上的代码中，06 行～08 行定义了 3 个元素。在第 2 个元素中，定义了绝对定位属性，同时定义了元素的大小等属性。在相邻的元素中，定义了元素的大小等属性。以上代码的显示效果如图 17-9 所示。

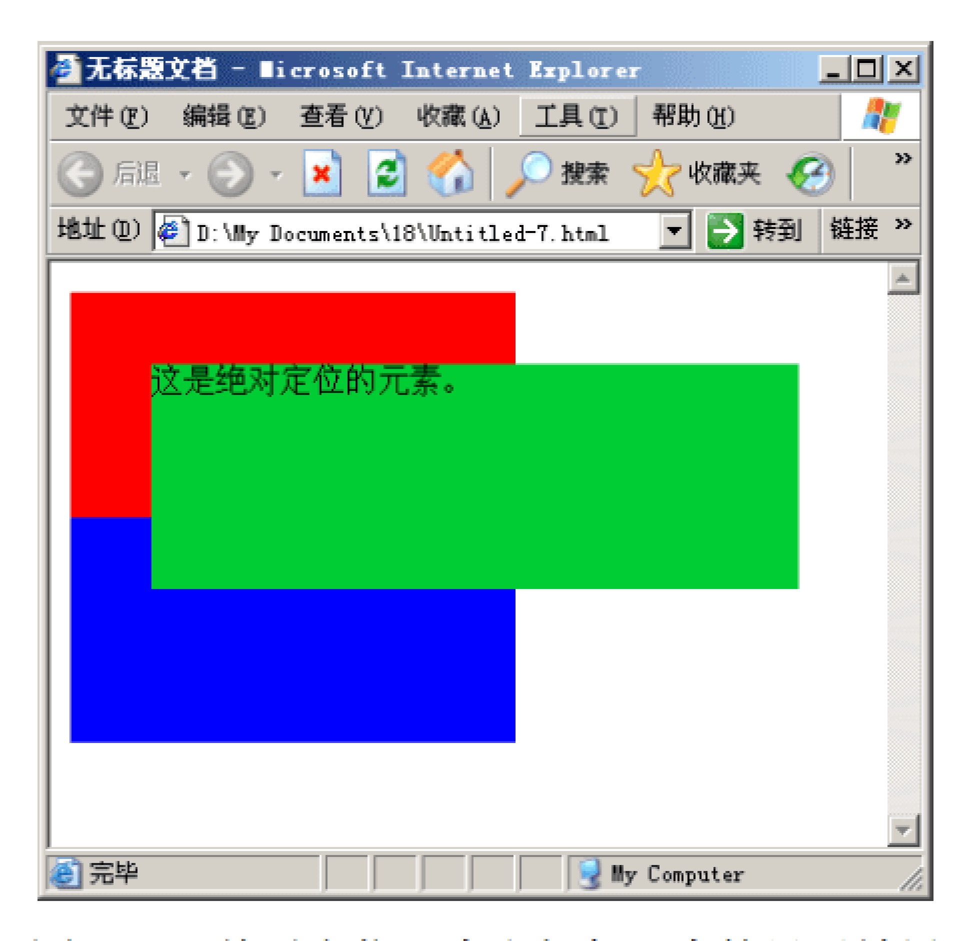


图 17-9　绝对定位元素和相邻元素的显示效果

从图 17-9 可以看出，与绝对定位元素相邻的元素的显示方式并不受绝对定位元素的影响。使用绝对定位的元素，其显示级别高于普通元素，所以绝对定位元素覆盖了普通页面元素。

17.3 相对定位

相对定位，是按照元素自身所在的位置，使用边偏移属性重新定义元素的显示位置。使用相对定位的元素，依然是文档中的元素。元素的显示位置和元素所在文档中其他元素相互关联，其具体内容如下所示。

17.3.1 相对定位元素位置的确定

在确定相对定位元素位置的时候，首先要确定元素的原是位置，即元素在文档中的位置。

然后根据边偏移属性中定义的偏移值，确定元素的最终位置。

下面是一个在元素中使用相对定位属性的实例，其代码如下所示。

例程 17-8 example.html

```
01 <html xmlns="http://www.w3.org/1999/xhtml">
02 <head>
03 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
04 <title>无标题文档</title>
05 <style>
06 .div1{
   background:red;
   width:200px;
   height:100px;
   }
07 .relative{
   position:relative;
   top:50px;
   left:100px;
   width:300px;
   height:100px;
   background:blue;}
08 </style>
09 </head>
10 <body>
   <div class="div1"></div>
   <div class="relative">这是相对定位的元素。</div>
11 </body>
12 </html>
```

在以上的代码中，06 和 07 行定义两个元素。其中第一个元素中，定义元素的背景颜色属性值为深灰色，大小为 200px×100px。在第 2 个元素中，定义元素定位属性值为相对定位，同时定义了元素的大小和背景等属性。以上代码的显示效果如图 17-10 所示。

从图 17-10 可以看出，使用相对定位的元素，元素按照自身所在的位置进行偏移。

图 17-10　相对定位属性的显示效果

17.3.2　相对定位与相邻元素

在使用相对定位的时候，相对定位元素保留原来所占有的空间，同时将自身按照边偏移属性中定义的属性值进行偏移(有可能覆盖其他元素)。与相对定位元素相邻的元素，会按照相对定位元素为普通元素的方式排列。

下面是一个相对定位与相邻元素的实例，其代码如下所示。

例程 17-9 example.html

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
```

```
03  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
04  <title>无标题文档</title>
05  <style>
06  .div1{
    background:red;
    width:200px;
    height:100px;
  }
07  .relative{
    position:relative;
    top:50px;
    left:100px;
    width:300px;
    height:100px;
    background:blue;}
08  .div2{
    background:yellow;
    width:200px;
    height:100px;
  }
09  </style>
10  </head>
11  <body>
    <div class="div1"></div>
    <div class="relative">这是相对定位的元素。</div>
    <div class="div2"></div>
12  </body>
13  </html>
```

在以上的代码中，06～08 行定义了 3 个元素。在第 2 个元素中，定义了相对定位属性，同时定义了元素的大小等属性。在相邻的元素中，定义了元素的大小等属性。以上代码的显示效果如图 17-11 所示。

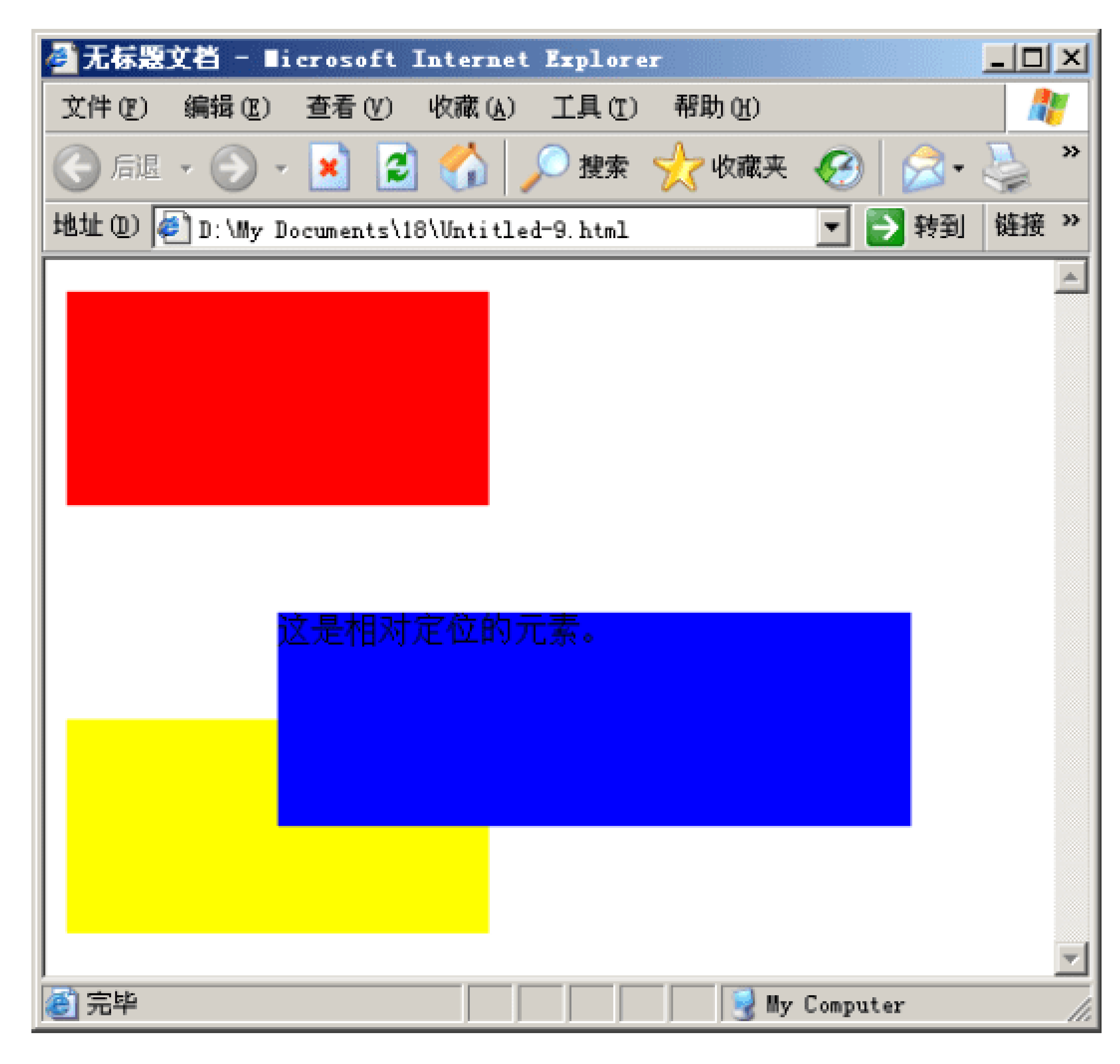


图 17-11　相对定位元素和相邻元素的显示效果

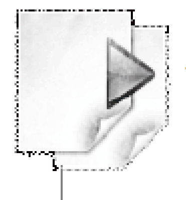
从图 17-11 可以看出，与相对定位元素相邻的元素的显示方式，会保留相对元素原来占有的空间。使用相对定义的元素，其显示级别高于普通元素，所以相对定位元素覆盖了普通页面元素。

17.4 定位元素的重叠

在一个页面中，如果使用几个定位元素，就可能发生定位元素重叠的情况。默认的情况下，后添加的元素会覆盖先添加的元素。通过使用层叠定位属性(z-index)，可以调整各个层叠元素的显示顺序，下面进行详细讲解。

层叠定位属性(z-index)用来定义定位元素的显示顺序。在层叠定位属性中，属性值使用 auto 值和没有单位的数字，其语法结构如下所示。

```
z-index : auto | number
```



注意

在非定位元素中，使用层叠定位属性，无法更改元素的显示层次。

下面是一个使用层叠定位属性的实例，其代码如下所示。

例程 17-10 z-index.html

```
01  <html xmlns="http://www.w3.org/1999/xhtml">
02  <head>
03  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
04  <title>无标题文档</title>
05  <style>
06  .index1{
    top:50px;
    left:50px;
    background:red;
    z-index:2;
  }
07  .index2{
    top:100px;
    left:100px;
    background:green;
    z-index:-1;
}
08  .index3{
    top:150px;
    left:150px;
    background:yellow;
    z-index:1;
```

```
    }
09  div{
    position:absolute;
    width:200px;
    height:150px;
}
10  </style>
11  </head>
12  <body>
    <div class="index1"></div>
    <div class="index2"></div>
    <div class="index3"></div>
13  </body>
14  </html>
```

以上的代码中，09 行定义了 3 个大小均为 200px×150px 的绝对定位元素。在第 1 个元素中，定义上边偏移属性和左边偏移属性值为 50px，背景颜色为红色，层叠顺序为 2。在第 2 个元素中，定义上边偏移属性和左边偏移属性值为 100px，背景颜色为绿色，层叠顺序为－1。在第 3 个元素中，定义上边偏移属性和左边偏移属性值为 150px，背景颜色为黄色，层叠顺序为 1。以上代码的显示效果如图 17-12 所示。如果取消层叠定位属性，则显示效果如图 17-13 所示。

从图 17-12 和图 17-13 可以看出，通过定义层叠定位属性，可以随意更改元素的显示顺序。

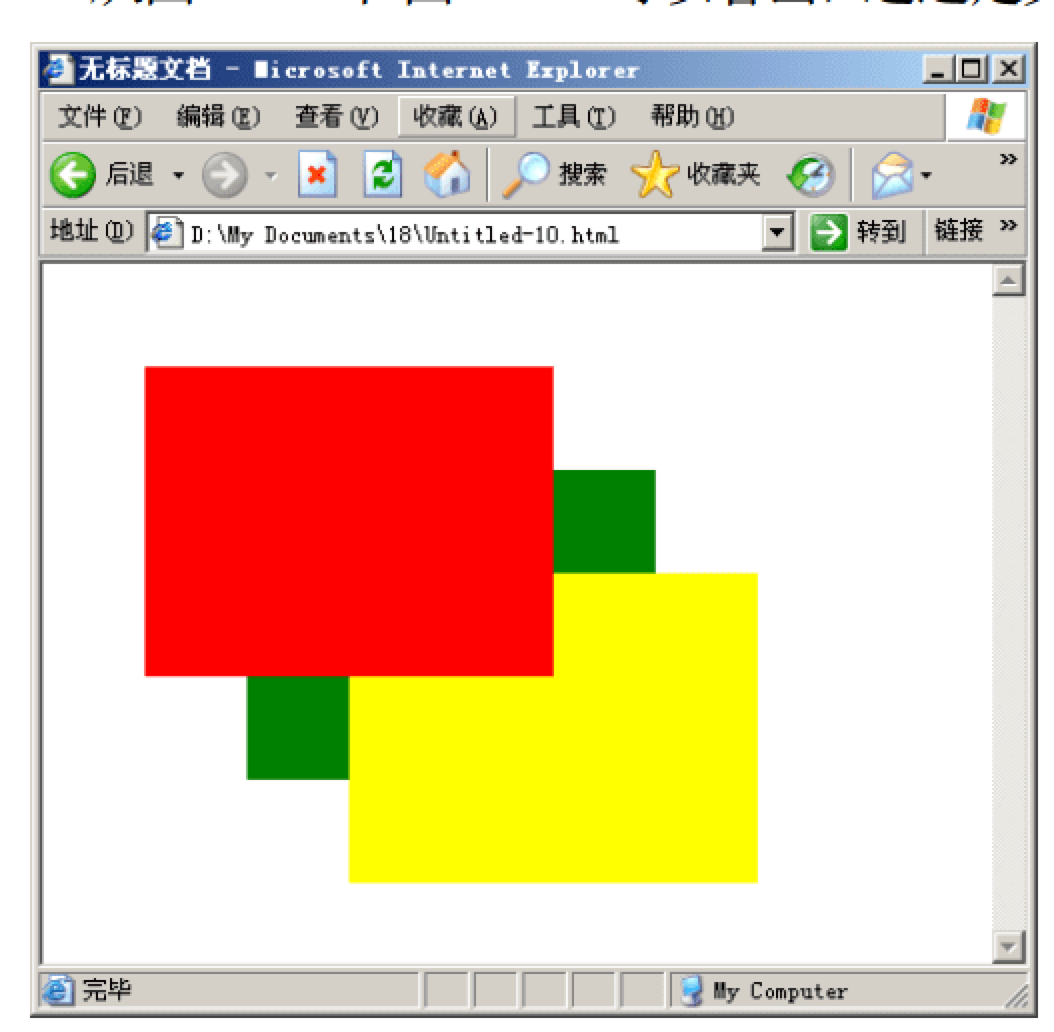


图 17-12　使用层叠定位属性的显示效果

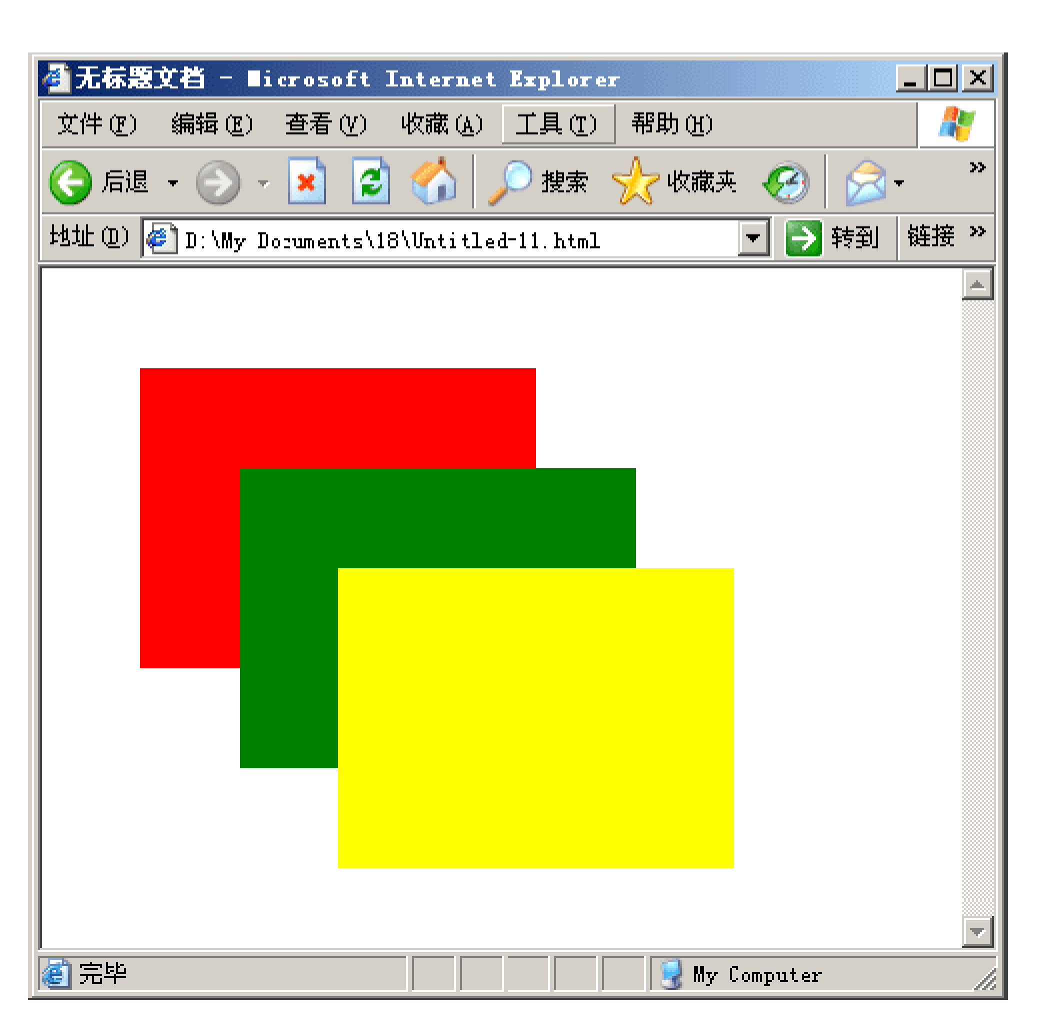


图 17-13　取消层叠属性的显示效果

17.5 本章习题

一、选择题

1．以下关于元素的定位属性(position)的说法中，错误的是：(　　)

A. 使用 static 元素按照自身默认的方式显示。

B. 使用 absolute，元素将不会有任何偏移。

C. 浏览器不支持 fixed 属性。。

D. 使用 relative 属性，元素以自身位置为基准，按照边偏移属性中定义的属性值显示。

2．以下有关边偏移属性的定义中，正确的是：(　　)

A. 边偏移属性中不可以使用百分比值。

B. 边偏移属性中使用百分比值的时候，只有在定义元素定位属性值为绝对定位(absolute)或者相对定位(relative)的时候，才能正常显示

C. Firefox 浏览器不支持边偏移属性。

D. 浏览器不支持左边偏移属性

3．以下有关使用绝对定位的使用中，错误的是：(　　)

A. 使用绝对定位的时候，应包含定位的参照元素。

B. 使用绝对定位的元素，会从文档中独立地显示出来，可能会遮盖其他的页面元素。

C. 在使用绝对定位的时候，不用参照元素，也会正常显示。

D. 在使用绝对定位的时候，绝对定位元素会独立显示，并不影响其他元素的显示位置。

二、填空题

1. 定位属性中，可以使用 4 种属性值，分别为________、________、________、________。

2. 边偏移属性包括________、________、________和________。

3. 层叠定位属性(z-index)、用来定义________________。属性值使用______和________。

三、实战练习

1．制作一个页面，右边偏移属性值分别设置为长度和百分比值，注意显示效果。

2．制作一个 3 层的页面，练习使用层叠定位属性。

Chapter 18

CSS控制元素的布局

在使用 CSS 布局页面的时候，通常使用布局属性进行来构建页面的框架和各个具体的内容部分。合理地使用布局属性，可以使页面的显示更加灵活。在 CSS 中，布局属性包括：浮动属性(float)、清除浮动属性(clear)、剪切属性(clip)、溢出属性(overflow)、显示方式属性(display)和可视属性(visibility)。

本章主要内容有:

◎ 理解 CSS 中，各个布局属性的特点和用法。
◎ 重点掌握浮动属性的作用和用法。
◎ 能够熟悉应用各个布局属性。
◎ 能够熟练知道每个布局属性在代码中的作用。

18.1 元素的浮动

在网页中，使用浮动属性可以更改块元素的默认显示方式，将原本换行显示的块元素同行显示。通过使用浮动元素和盒模型中的各种属性，可以对各种页面元素进行布局，下面进行详细讲解。

18.1.1　元素的浮动属性 float

元素的浮动属性(float)用来定义块元素的浮动方式。在浮动属性中，可以使用 4 种属性值，分别为 static、absolute、fixed 和 relative，其语法结构如下所示。

```
float : none | left | right;
```

其中每个属性值的含义，如下所示。

- none：元素不浮动。
- left：元素浮动在左侧。
- right：元素浮动在右侧。

下面是一个使用元素浮动属性的实例，其代码如下所示。

例程 18-1　float.html

```
01  <html xmlns="http://www.w3.org/1999/xhtml">
02  <head>
03  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
04  <title>使用浮动属性的实例</title>
05  <style>
06  div{
    float:right;
    width:300px;
    height:200px;
    background:#FFFFFF;
    border:4px solid #000000;
  }
07  </style>
08  </head>
09  <body>
10  <div>驿外断桥边，寂寞开无主。已是黄昏独自愁，更著风和雨。
    无意苦争春，一任群芳妒。零落成泥碾作尘，只有香如故。</div>
11  </body>
12  </html>
```

以上的代码中，06 行中定义了元素的浮动属性值为 right，同时定义元素的大小为 300px×200px，背景颜色为淡黄色，边框为 4 像素黑色实线边框。此时元素将显示在浏览器的右侧，

其显示效果如图 18-1 所示。

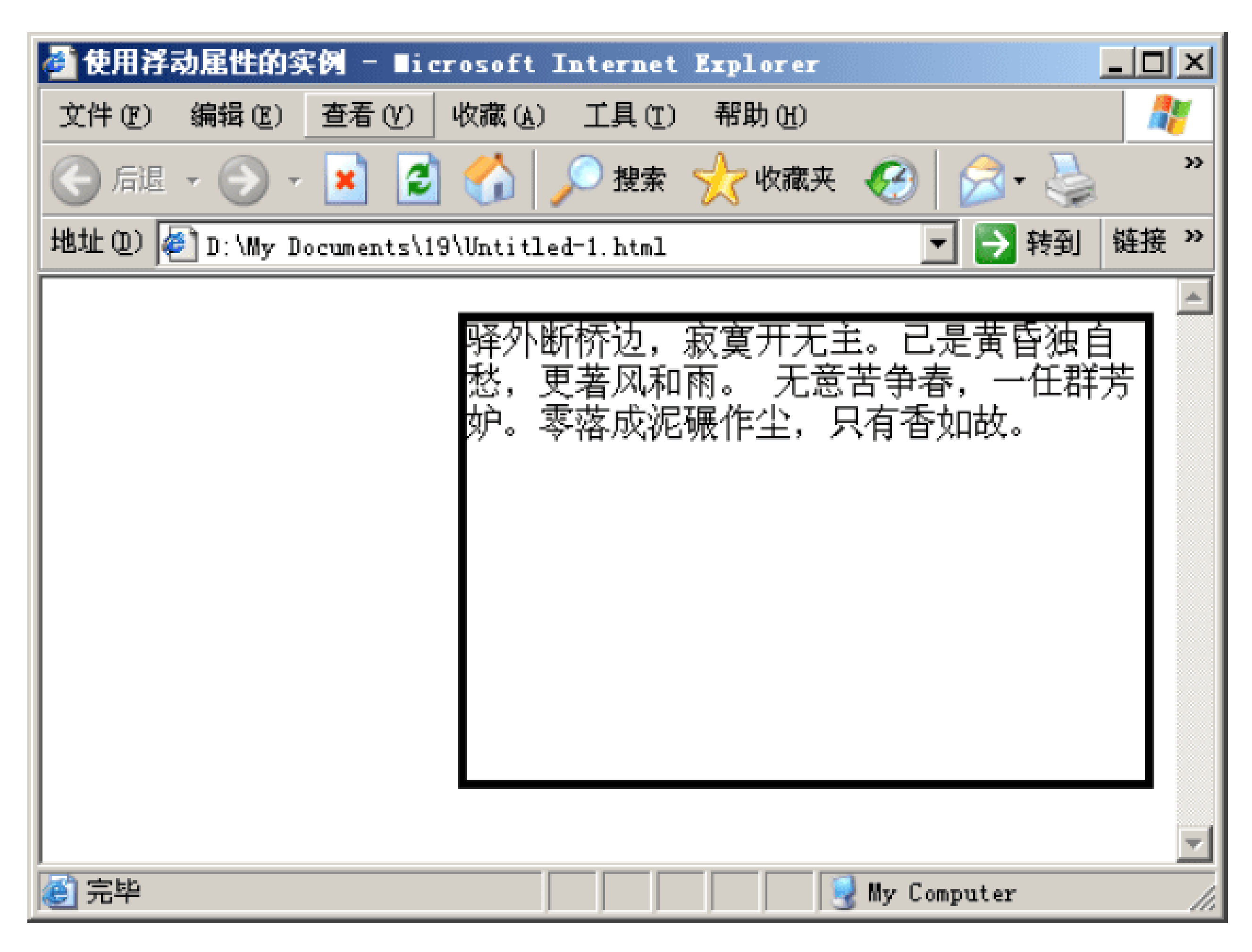


图 18-1　浮动属性的显示效果

18.1.2　浮动元素和固定元素

在制作页面的时候，如果相邻的两个元素中一个为浮动元素，另一个为固定元素。则最终的显示效果，和元素之间的位置有关。如果浮动元素处于固定元素之前，则固定元素和浮动元素同行显示(是否同行显示还和父元素的宽度有关，这里假定父元素足够宽)。如果浮动元素处于固定元素之后，则浮动元素换行显示。

下面是一个使用浮动元素和固定元素的实例，其代码如下所示。

例程 18-2　example.html

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>无标题文档</title>
<style>
.float{
float:left;
width:200px;
height:100px;
background:red;
}
.nofloat{
width:300px;
height:100px;
background:blue;
}
.float2{
background:green;
}
```

```
09 </style>
10 </head>
11 <body>
   <div class="float">这里使用了浮动元素</div>
   <div class="nofloat">这里使用了固定元素</div>
   <div class="float float2">这里使用了浮动元素</div>
12 </body>
13 </html>
```

以上的代码中，06～08 行中分别定义了 3 个元素。其中，第 1 个和第 3 个为浮动元素，第 2 个为固定元素。在每个元素中，都定义了相应的大小和背景属性。此时，第 1 和第 2 个元素将同行显示，第 3 个元素将换行显示。以上代码的显示效果如图 18-2 所示。

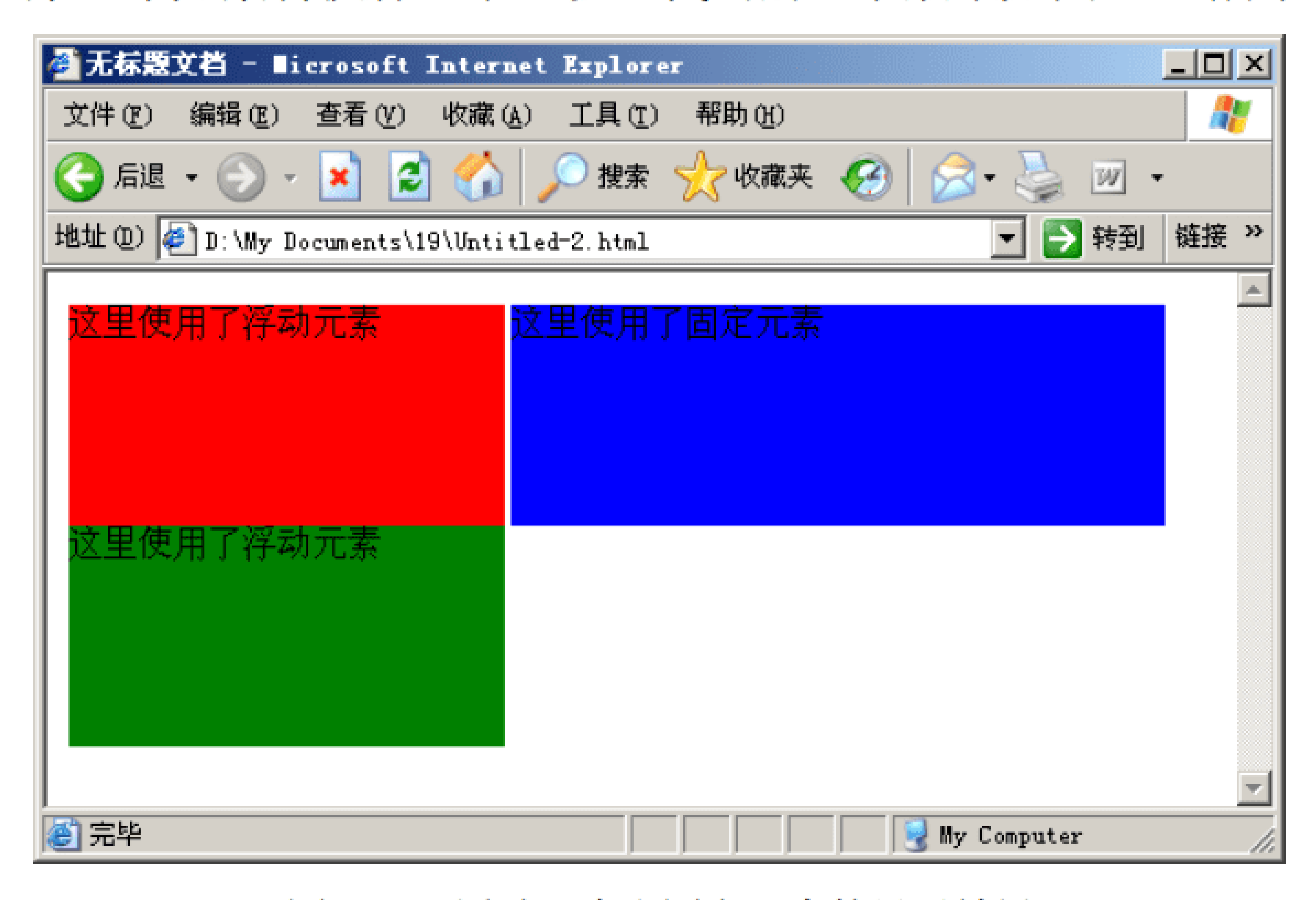


图 18-2　浮动元素和固定元素的显示效果

18.1.3　两个浮动元素的显示效果

在制作页面的时候，相邻的两个浮动元素，子父元素足够大的情况下，会同行显示。如果两个元素中定义的浮动属性值不同，则两个元素分别显示在父元素的两侧，中间空白区域显示父元素的背景颜色或图像。

下面是一个使用两个浮动元素的实例，其代码如下所示。

例程 18-3　example.html

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>无标题文档</title>
<style>
.float{
float:left;
width:200px;
height:100px;
background:red;
```

```
        }
07      .float2{
        float:right;
        width:200px;
        height:100px;
        background:blue;
      }
08      </style>
09      </head>
10      <body>
        <div class="float">这里使用了浮动元素</div>
        <div class="float2">这里也使用了浮动元素</div>
11      </body>
12      </html>
```

以上的代码中，06 行和 07 行中定义了两个浮动元素。其中，第 1 个浮动元素中定义浮动属性值为左侧，第 2 个浮动元素中定义浮动属性值为右侧。以上代码的显示效果如图 18-3 所示。

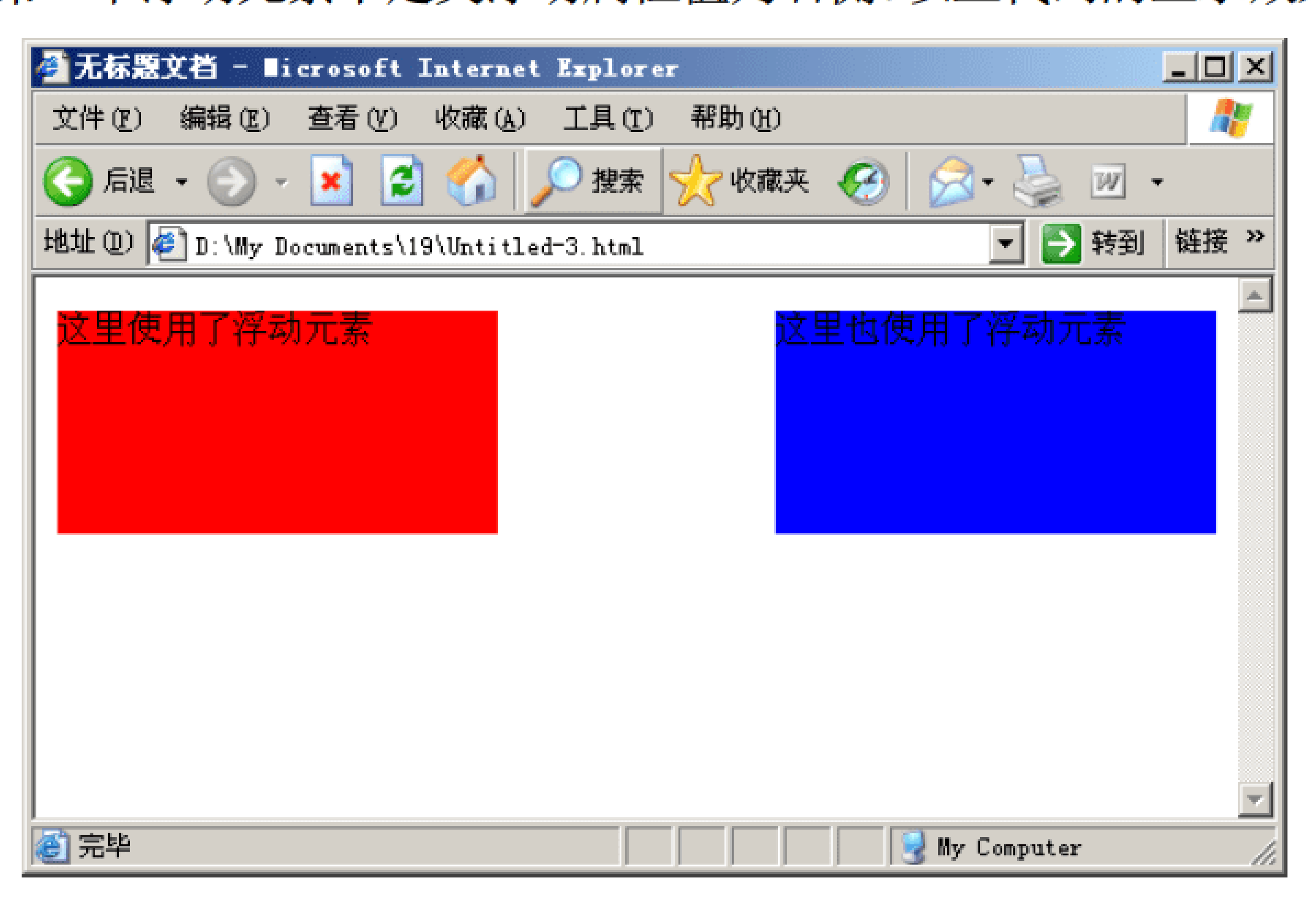


图 18-3　两个浮动元素的显示效果

18.1.4　多个浮动元素的显示顺序

在制作页面的时候，如果同时使用多个浮动元素，页面会按照浮动元素定义的先后对元素进行排列。后定义的元素默认地排列在先定义元素的旁边。

下面是一个使用多个浮动元素的实例，其代码如下所示。

例程 18-4　example.html

```
01      <html xmlns="http://www.w3.org/1999/xhtml">
02      <head>
03      <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
04      <title>无标题文档</title>
05      <style>
06      .float1
```

```
{
    float:left;
    background:#cccccc;
}
07  .float2
{
    float:right;
    background:#333333;
}
08  .float3
{
    float:left;
    background:#666666;
}
09  .float4
{
    float:right;
    background:#999999;
}
10  div
{
    width:150px;
    height:75px;
}
11  </style>
12  </head>
13  <body>
    <div class="float1">注意页面的显示效果</div>
    <div class="float2"></div>
    <div class="float3">注意页面的显示效果</div>
    <div class="float4"></div>
13  </body></html>
```

以上的代码中，06 行～09 行定义了 4 个浮动元素。其中，第 1 个和第 3 个浮动元素中，定义浮动属性值为左侧，第 2 个和第 4 个浮动元素中，定义浮动属性值为右侧。同时定义 4 个元素的背景颜色为不同的灰色，以上代码的显示效果如图 18-4 所示。

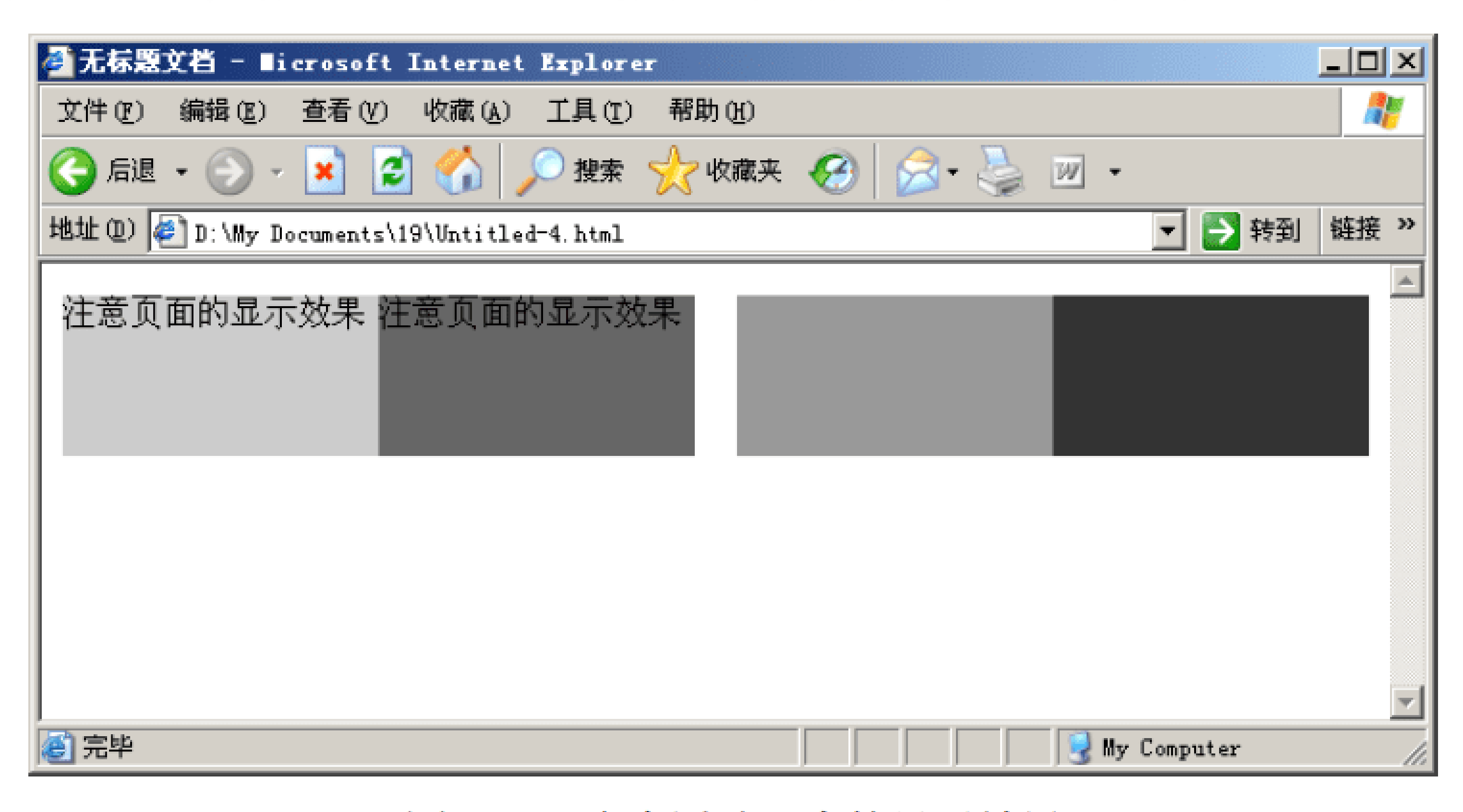


图 18-4　多个浮动元素的显示效果

从图 18-4 可以看出，第 1 个浮动元素，由于最优先定义，所以会显示在最左侧。第 2 个浮动元素，由于浮动的位置不同，所以显示到最右侧。第 3 个浮动元素，由于第 1 个元素的存在，所以无法浮动到最左侧，会显示在第 1 个浮动元素的右面。第 4 个浮动元素的情形和第 3 个浮动元素类似，所以会显示在第 2 个浮动元素的左侧。

18.2 浮动的清除

在网页中使用浮动元素的时候，会影响后面相邻的固定元素。由于在不同的浏览器中，对浮动属性的解释存在差异，所以需要对某些浮动的属性进行清除。清除浮动元素时，使用的属性是清除浮动属性(clear)，下面进行详细讲解。

18.2.1 清除浮动属性 clear

清除浮动属性(clear)用来清除与元素相邻的浮动元素。在清除浮动属性 clear 中，可以使用 4 种属性值，分别为 none、left、right、both，其语法结构如下所示。

```
clear : none | left | right | both;
```

其中每个属性值的含义，如下所示。

- none：不清除浮动。
- left：清除元素左侧的浮动元素。
- right：清除元素右侧的浮动元素。
- both：清除元素两侧的浮动元素。

下面是一个使用清除浮动属性的实例，其代码如下所示。

例程 18-5 clear.html

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>无标题文档</title>
<style>
.float
{
    float:left;
    width:300px;
    height:100px;
    background:red;
}
.clear
{
    clear:left;
    float:left;
```

```
        width:200px;
        height:100px;
        background:blue;
    }
08  </style>
09  </head>
10  <body>
    <div class="float">这里使用了浮动元素</div>
    <div class="clear">这里清除了浮动属性</div>
11  </body>
12  </html>
```

以上的代码中，06 行和 07 行定义了两个浮动元素。在第 1 个浮动元素中，定义元素的浮动属性值为左浮动，并定义了元素的大小和背景属性，便于元素的显示。在第 2 个浮动元素中，定义元素的浮动属性值为左浮动，清除浮动属性值为清除左侧浮动，同时定义了元素的大小和背景属性。此时由于清除了左侧浮动元素，所以两个元素将换行显示，其显示效果如图 18-5 所示。

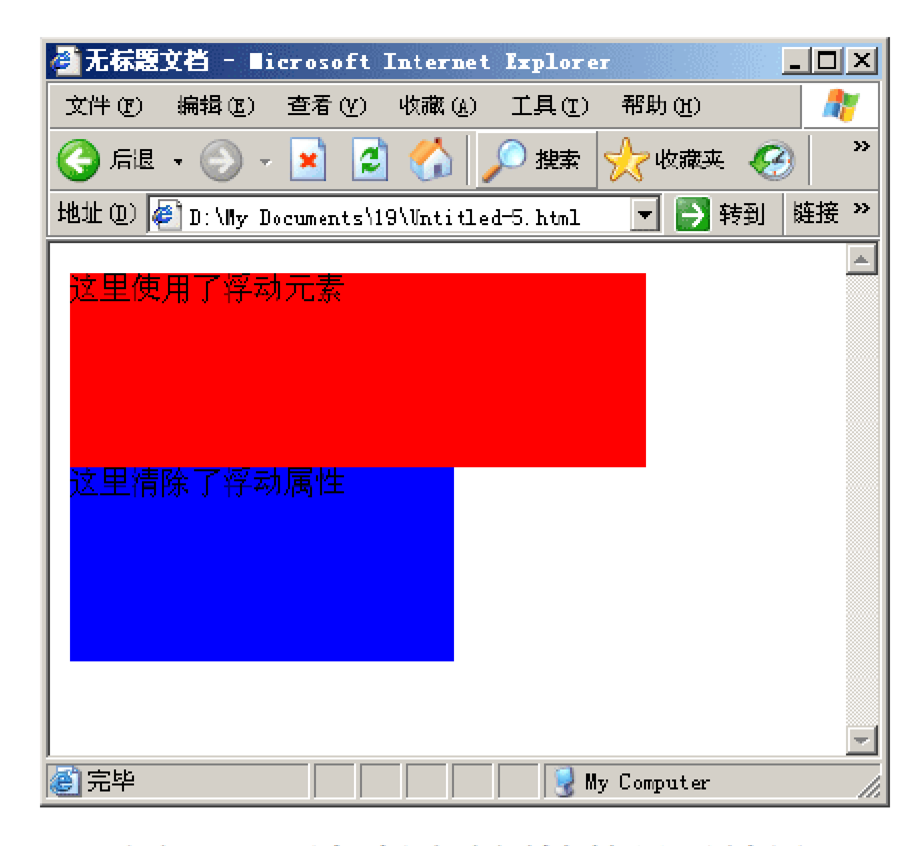


图 18-5　清除浮动属性的显示效果

18.2.2　清除浮动与固定元素

当清除浮动元素右侧的浮动元素的时候，由于固定元素在浮动元素的后面时，会同行显示，所以无法显示清除浮动的效果。在实际使用清除浮动属性的时候，常常会忽略这一点，造成布局上的困扰。

下面是一个清除右侧浮动元素的实例，其代码如下所示。

例程 18-6　example.html

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>无标题文档</title>
<style>
.float
{
    float:left;
    width:200px;
    height:100px;
    background:red;
}
.clear
{
    clear:right;
    float:left;
```

```
        width:200px;
        height:100px;
        background:blue;
    }
    </style>
    </head>
    <body>
    <div class="clear">how are you</div>
    <div class="float">how are you</div>
    </body>
    </html>
```

以上的代码中，定义了两个浮动元素。在第 1 个浮动元素中，定义元素的浮动属性值为左浮动，清除浮动属性值为清除右侧浮动。在第 2 个浮动元素中，定义元素的浮动属性值为左浮动。同时在两个元素中都定义了元素的大小和背景属性。此时由于右侧元素即使没有浮动属性，依然可以显示在第 1 个元素的右侧，所以无法显示清除浮动的效果，其显示效果如图 18-6 所示。

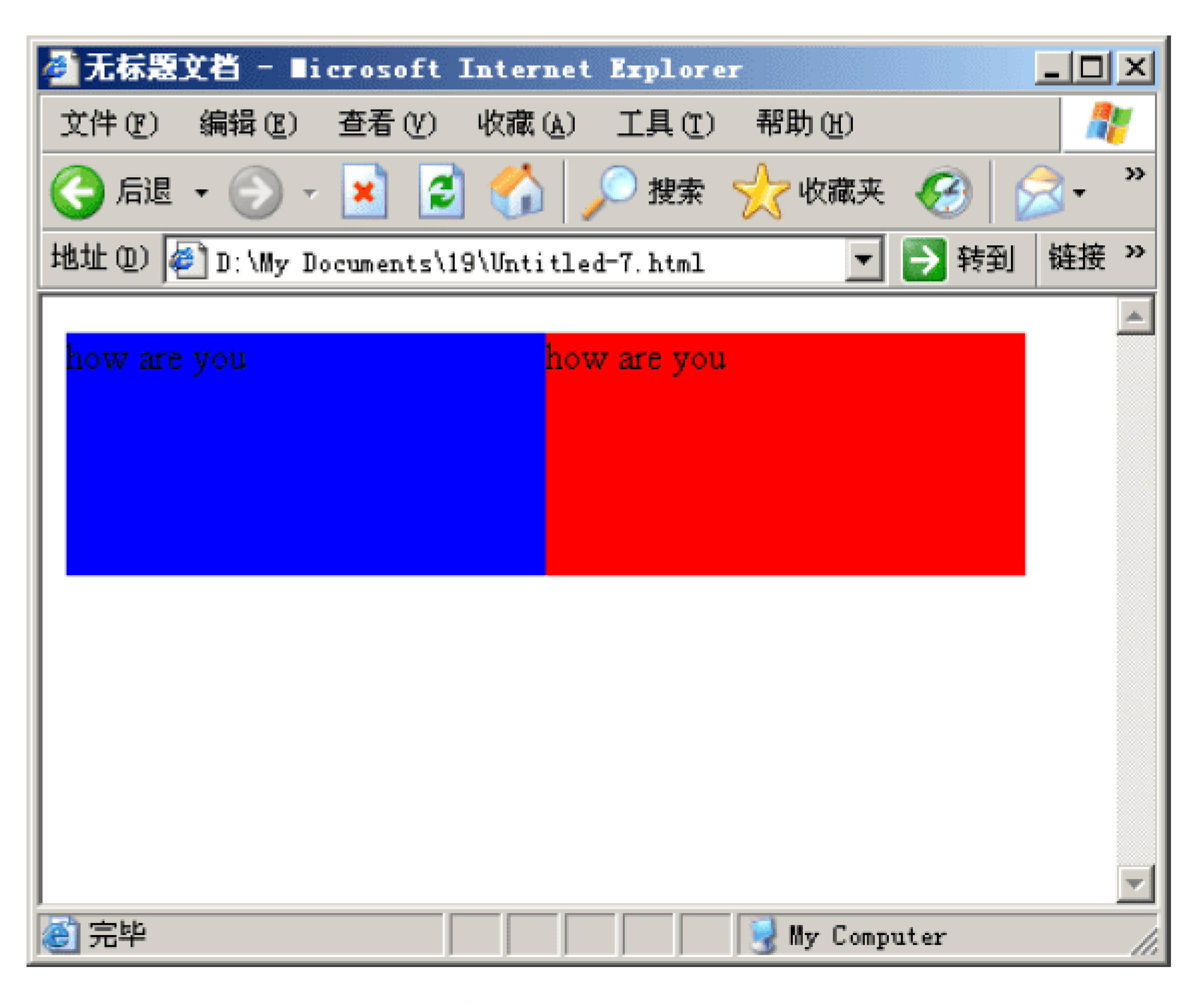


图 18-6　清除右侧浮动元素的显示效果

18.3 内容的剪切

在 CSS 中，可以使用剪切属性，对元素内容的可视化区域进行控制。剪切区域所使用的属性是 clip 属性，其中元素属性值的写法和其他的属性略有不同，下面进行详细讲解。

18.3.1 内容的剪切属性 clip

内容的剪切属性(clip)用来裁减元素的可视化范围。在内容的剪切属性，可以使用两种属性值，一种为 auto 值，另一种为区域值，其语法结构如下所示。

```
clip : auto | rect ( number number number number );
```

其中 rect 中定义的 4 个属性值，是以元素左上角为中心，按照上、右、下、左的顺序定义剪切区域的四条边线，在四条边线划定的区域，将显示元素内容。

注意

内容的剪切属性(clip)只能使用在绝对定位的元素中，如果未定义元素为绝对定位，则无法显示剪切效果。

下面是一个使用内容的剪切属性的实例，其代码如下所示。

例程 18-7 clip.html

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>无标题文档</title>
<style>
.clip
{
    position:absolute;
    top:50px;
    left:50px;
    clip:rect(20px 140px 40px 80px)
    width:200px;
    height:100px;
    background:red;
}
.main
{
    width:300px;
    height:300px;
    background:#cccccc
}
</style>
</head>
<body>
    <div class="main">
    <div class="clip">注意出现剪切的位置</div></div>
</body>
</html>
```

以上的代码中，定义了两个嵌套的元素。在子元素中定义了元素的剪切属性，同时定义了元素的大小和背景属性，并定义元素为绝对定位。在父元素中，定义了元素的背景和大小等属性。此时，在子元素中，只有被定义剪切的区域内的部分才能够显示，其他部分以透明的方式显示，其显示效果如图 18-7 所示。

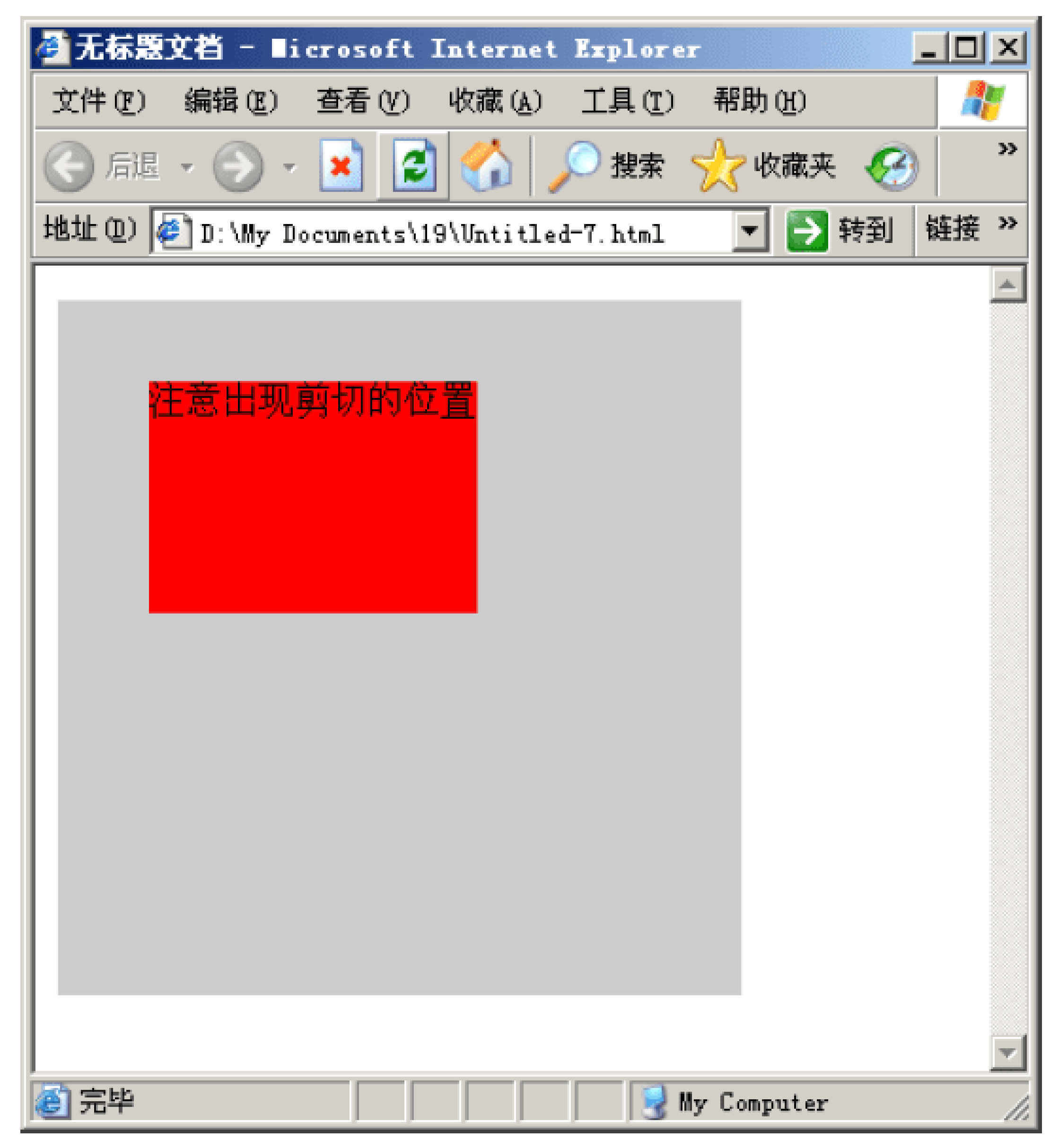


图 18-7　定义剪切属性后的显示效果

18.3.2　剪切属性与内容

在使用剪切属性的时候，元素内容的显示方式并不发生改变。元素中的内容，在剪切区域之外的部分将会消失，但是元素占有的空间并不发生改变。

下面是一个剪切元素中内容的实例，其代码如下所示。

例程 18-8　example.html

```
01  <html xmlns="http://www.w3.org/1999/xhtml">
02  <head>
03  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
04  <title>无标题文档</title>
05  <style>
06  .clip
{
    position:absolute;
    top:20px;
    left:20px;
    clip:rect(5px 140px 80px 20px);
    width:300px;
    height:300px;
    background:red;
}
07  .main
{
    width:300px;
    height:300px;
```

```
        background:#000000;
    }
</style>
</head>
<body>
<div class="main">
<div class="clip">这里是剪切元素中的内容，注意内容被裁剪的情况。</div></div>
</body>
</html>
```

以上的代码中，定义了两个嵌套的元素。在子元素中定义了元素的剪切属性，同时定义了元素的大小和背景属性，并定义元素为绝对定位。在父元素中，定义了元素的背景和大小等属性。在子元素中存在部分文本内容，当使用剪切属性后，部分文本内容和背景将会消失。但是原有文本内容的所在位置不发生变化。以上代码的显示效果如图 18-8 所示。如果取消剪切属性，则页面的显示效果如图 18-9 所示。

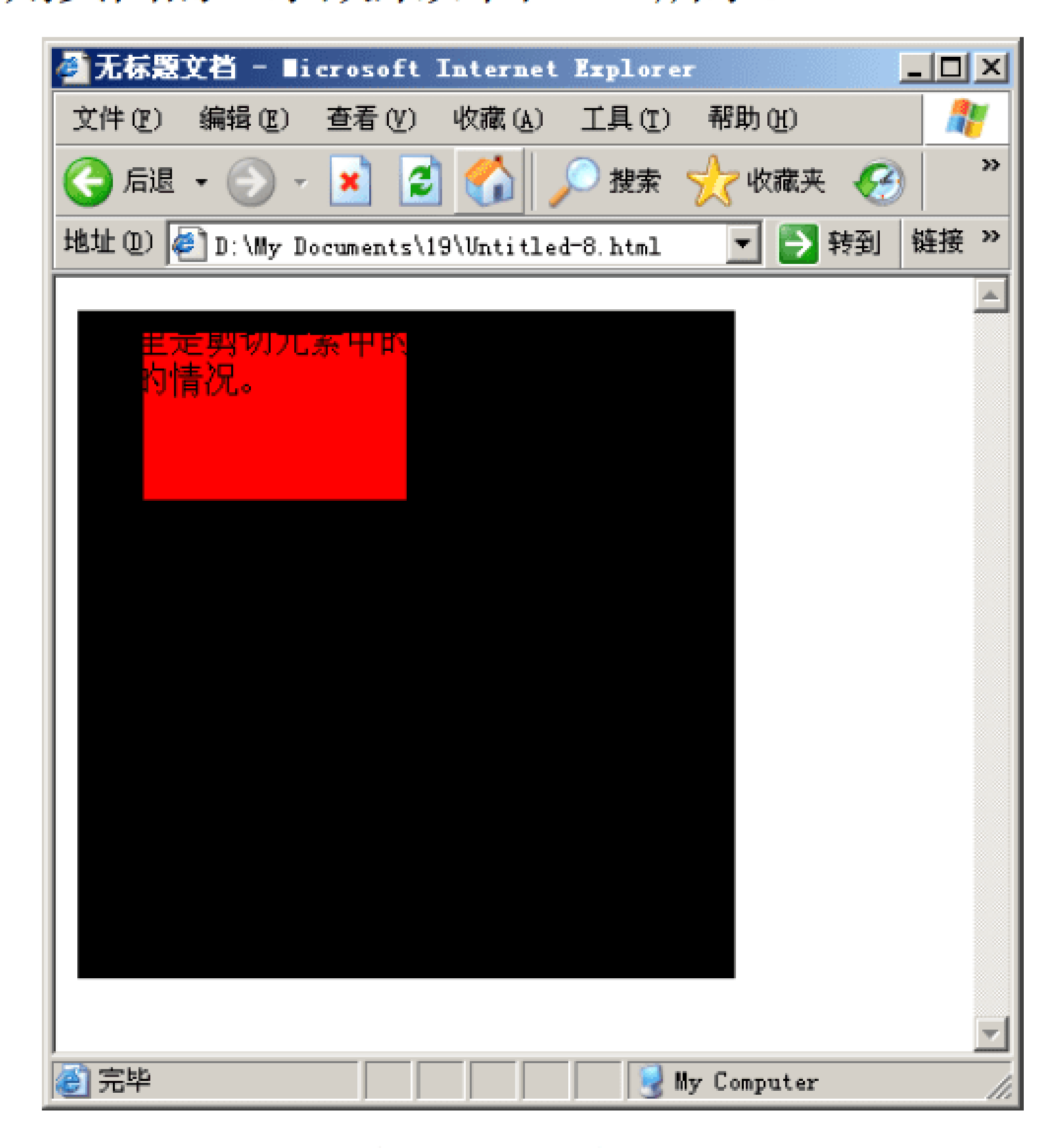


图 18-8　使用剪切属性后的显示效果

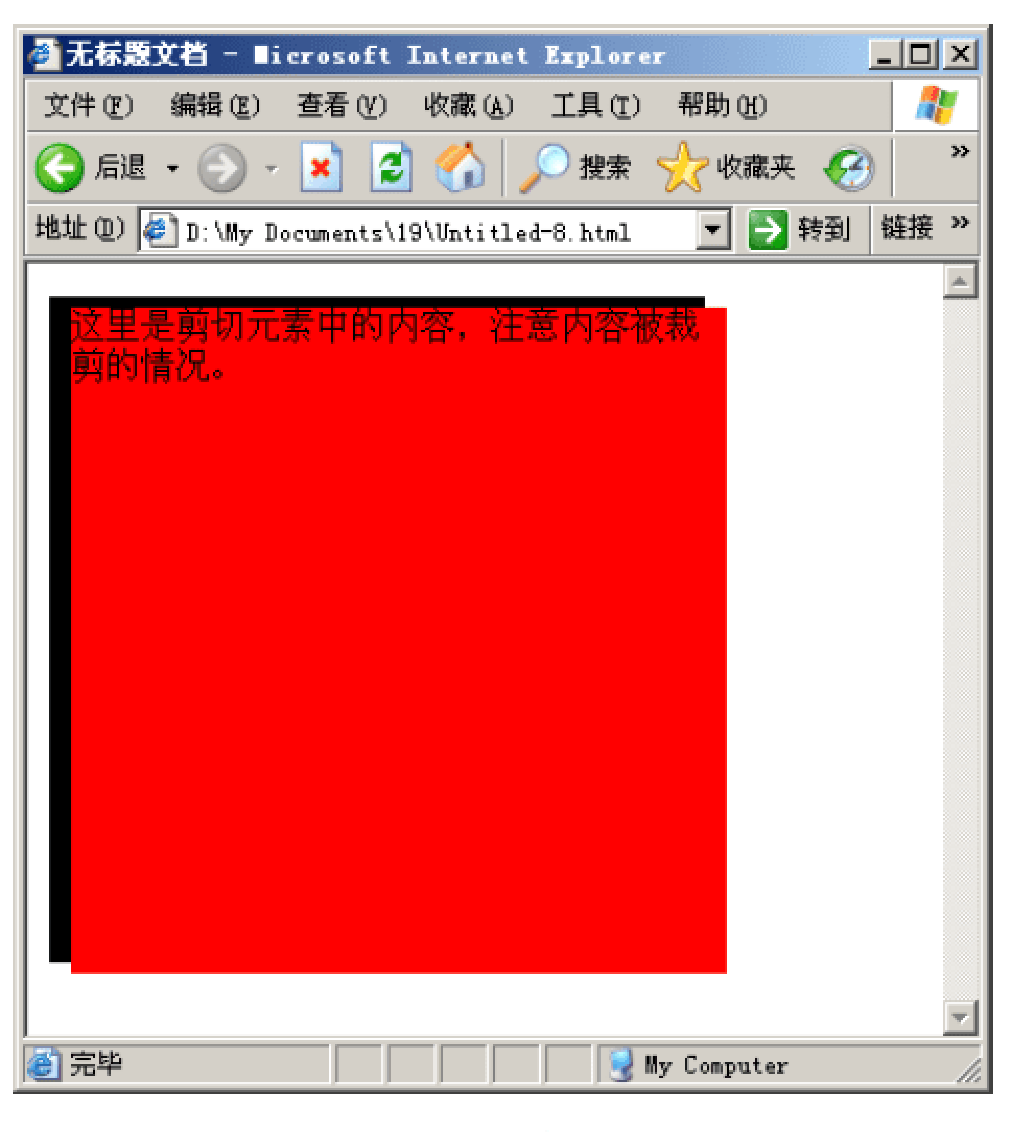


图 18-9　取消剪切属性后的显示效果

18.4 溢出内容的控制

在一个元素中，有时候会发生内容超出元素定义大小的情况。在 CSS 中，可以使用溢出属性(overflow)对溢出的内容，定义几种显示方式，如隐藏或者显示滚条等。使用溢出属性可以方便地控制溢出内容的显示，并能够保证元素的大小不被破坏，下面进行详细讲解。

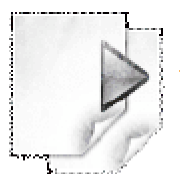
18.4.1 溢出属性 overflow

溢出属性(overflow)用来定义元素溢出部分的显示方式。在溢出属性中，可以使用 4 个属性值，分别为 visible、auto、hidden 和 scroll，其语法结构如下所示。

```
overflow : visible | auto | hidden | scroll;
```

其中每个属性值的含义如下所示。

- visible：默认属性，溢出内容按照原有的方式显示。
- auto：当内容超出元素定义的大小的时候，显示滚条，否则正常显示内容。
- hidden：隐藏溢出的内容。
- scroll：总是显示滚条。



注意

当在元素中使用了溢出属性时，元素中定义的剪切属性将会失效。

下面是一个使用溢出属性的实例，其代码如下所示。

例程 18-9 overflow.html

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>无标题文档</title>
<style>
div
{
    overflow:auto;
    width:200px;
    height:100px;
    background:#999999;

</style>
</head>
<body>
    <div>醉里挑灯看剑，梦回吹角连营。</div>
</body></html>
```

以上的代码中，在<div>元素中定义溢出属性值为 auto，同时定义元素的大小为 200px×100px。当元素中内容很少的时候，元素的显示效果如图 18-10 所示。当元素中内容超出元素大小的时候，元素的显示效果如图 18-11 所示。



图 18-10　内容小于元素大小的显示效果



图 18-11　内容大于元素大小的显示效果

18.4.2　横向溢出属性 overflow-x

横向溢出属性(overflow-x)用来定义元素溢出部分在水平方向上的显示方式。在横向溢出属性中，使用的属性值和溢出属性(overflow)的属性值完全相同，其语法结构如下所示。

```
Overflow-x: visible | auto | hidden | scroll;
```

注意

当在元素中使用了单向溢出属性时，元素中定义的剪切属性将会失效。同时另一个方向上，溢出属性将使用默认值。

下面是一个使用横向溢出属性的实例，其代码如下所示。

例程 18-10　overflow-x.html

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>无标题文档</title>
<style>
div
{
    overflow-x:auto;
    width:300px;
    height:100px;
    background:#999999;
}
</style>
```

```
</head>
<body>
<div><img src="pic.jpg" alt="pic"/></div>
</body>
</html>
```

以上的代码中，在<div>元素中定义横向溢出属性值为 auto，同时定义元素的大小为 300px×100px。在元素中定义了一个很大的图片，此时元素的横向上将会显示滚条。其代码运行后的显示效果如图 18-12 所示。

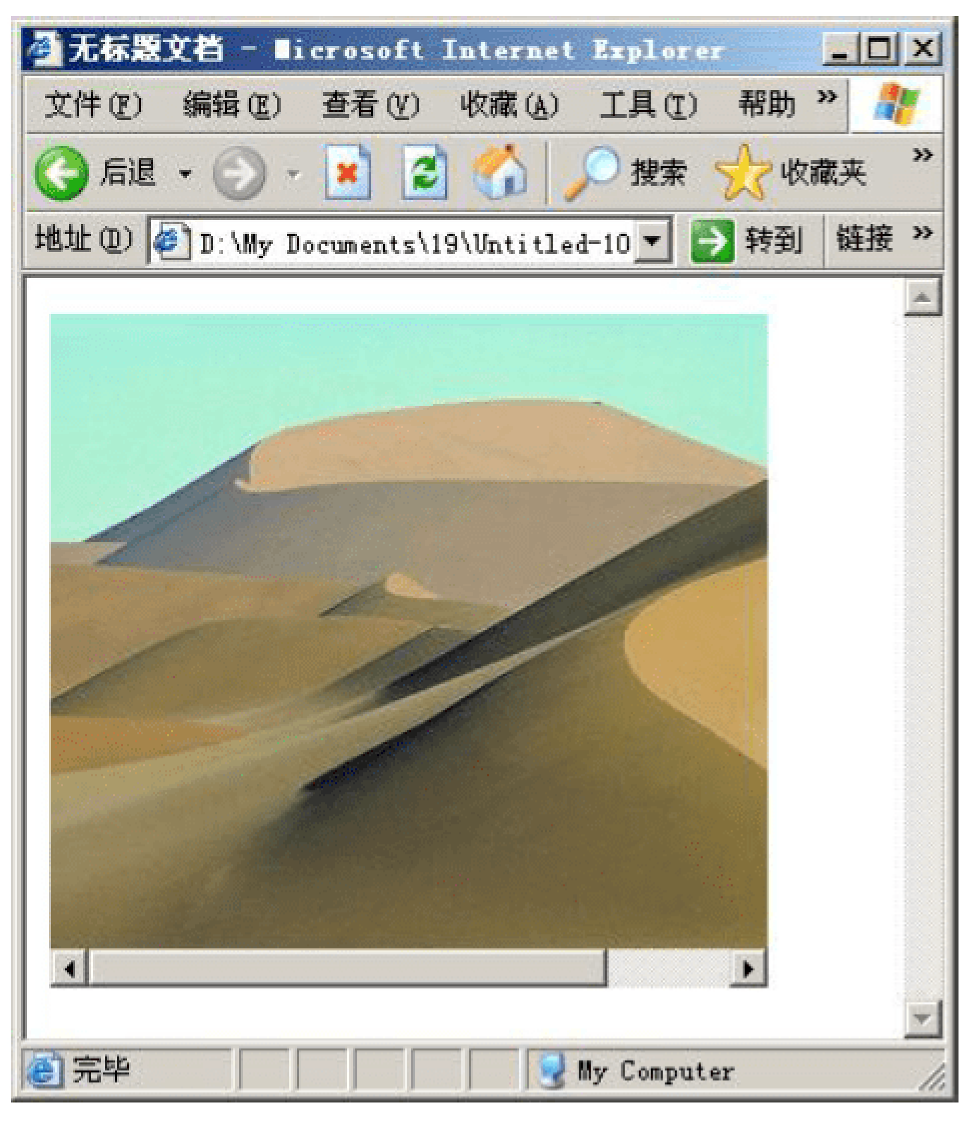


图 18-12 单向溢出属性的显示效果

18.4.3 纵向溢出属性 overflow-y

纵向溢出属性(overflow-y)用来定义元素溢出部分在垂直方向上的显示方式。在纵向溢出属性中，使用的属性值和溢出属性(overflow)的属性值完全相同，其语法结构如下所示。

```
overflow-y: visible | auto | hidden | scroll;
```

下面是一个使用纵向溢出属性的实例，其代码如下所示。

例程 18-11 overflow-y.html

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>无标题文档</title>
<style>
div
{
    overflow-y:auto;
    width:320px;
```

```
    height:210px;
    background:#999999;
}
</style>
</head>
<body>
<div><img src="pic.jpg" alt="pic"/></div>
</body>
</html>
```

以上的代码中，在<div>元素中定义纵向溢出属性值为 auto，同时定义元素的大小为 320px ×210px。在元素中定义了一个很大的图片，此时元素的纵向上将会显示滚条。其代码运行后的显示效果如图 18-13 所示。

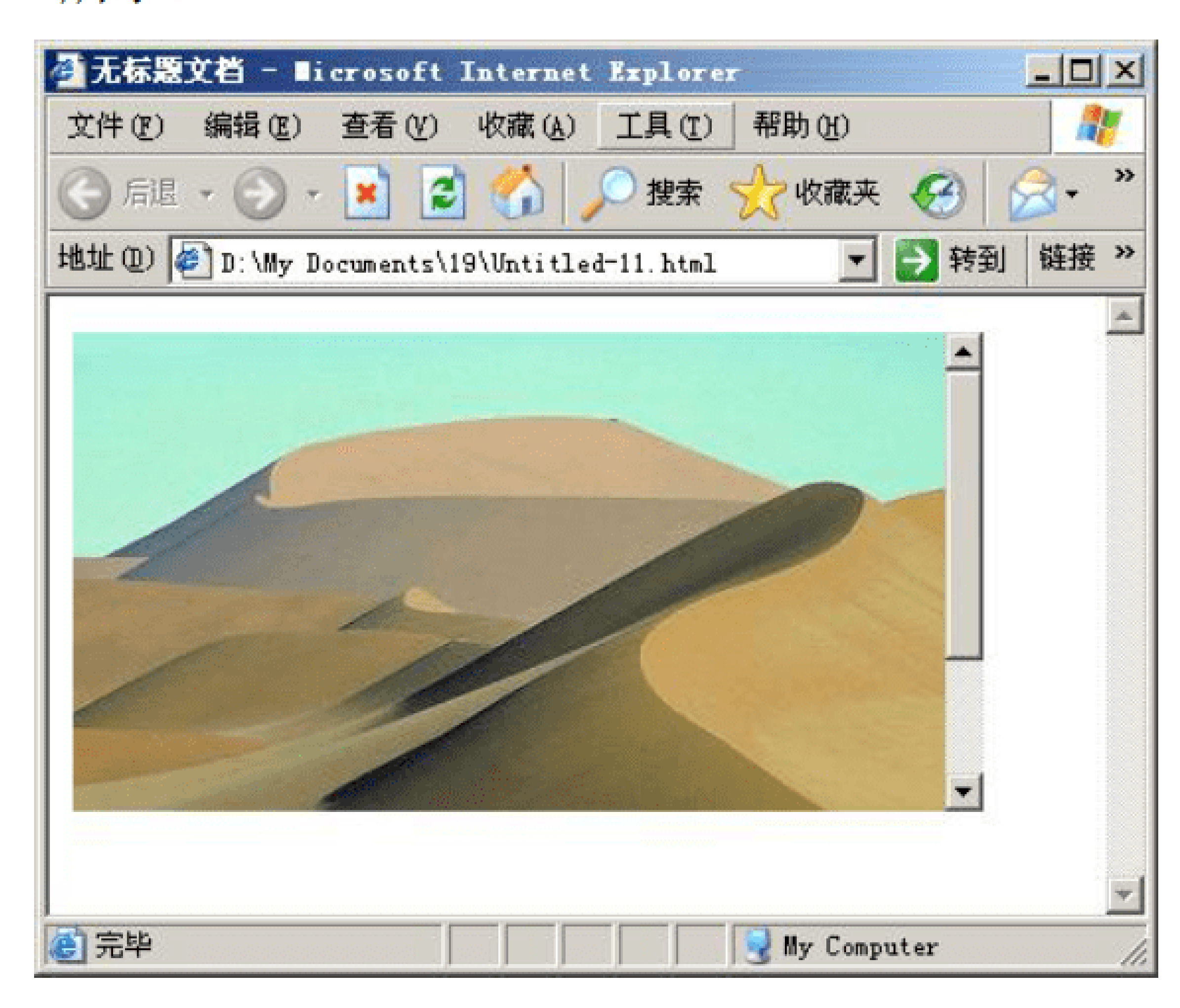


图 18-13 纵向溢出属性的显示效果

18.4.4 滚条和边框

在使用溢出属性的时候，如果定义元素显示滚条，同时定义了边框，要注意滚条和边框之间的关系。滚条的部分会显示在边框的内部，如果在滚条处不想显示边框，则可以通过定义单侧边框属性，取消滚条一侧的边框。

下面是一个显示边框和滚条关系的实例，其代码如下所示。

例程 18-12 example.html

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>无标题文档</title>
<style>
div
{
    overflow:auto;
```

```
        border:9px solid #000000;
        width:220px;
        height:100px;
        background:yellow;
    }
</style>
</head>
<body>
<div>这是关于溢出属性的实例，在本实例中使用了 auto 值，当内容部分超出元素大小的时候，将会在元素的右边显示滚条。注意观察滚条和边框之间的关系。</div>
</body>
</html>
```

以上的代码中，在元素中定义了溢出属性值为 auto，同时定义边框样式为 9 像素宽实线边框。此时，滚条部分内容，将显示在边框的内部，其显示效果如图 18-14 所示。



图 18-14　滚条与边框关系的显示效果

18.5 元素的显示方式

在 CSS 中，可以通过相应的属性控制元素的显示方式。例如，可以将原本以内联方式显示的元素，以块元素的方式显示。更改元素显示方式使用的是显示方式属性 display。通过使用显示方式属性，可以方便地更改元素默认的显示方式，制作出特殊的显示效果，如导航的图片转换效果等。下面进行详细讲解。

18.5.1 显示方式属性 display

显示方式属性(display)用来更改元素默认的显示方式。在显示方式属性中，可以使用的属性值有很多，其中部分属性值还不被浏览器所支持。其语法结构如下所示。

```
display : block | none | inline | compact | marker | inline-table | list-item | run-in | table | table-caption | table-cell |
```

```
table-column | table-column-group | table-footer-group | table-header-group | table-row | table-row-group;
```

其中部分属性值的含义如下所示。

- block：定义元素按照块元素的方式显示。
- none：定义元素隐藏。
- inline：定义元素按照内联元素的方式显示。
- inline-table：定义元素显示内联的方式，与相邻的内联元素同行显示。同时内容部分显示块元素的属性。
- list-item：定义元素以列表元素的方式显示。
- table-column-group：定义元素以表格标题组的方式显示，即在表格跨页的时候，在每页中都以表头的方式显示。
- table-footer-group：定义元素以表格标题组的方式显示，即在表格跨页的时候，在每页中都以表头的方式显示。

除了以上列举出来的各种属性值，其余的各个属性值，IE 浏览器还不支持(在其他浏览器中，有部分属性被支持)。

下面是一个显示方式属性的实例，其代码如下所示。

例程 18-13 display.html

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>无标题文档</title>
<style>
span
{
    display:block;
    width:200px;
    height:100px;
    background:#999999;
}
div
{
    display:inline;
    width:300px;
    height:100px;
    background:red;
}
p
{
    display:list-item;
    width:300px;
    height:100px;
    background:yellow;
}
img
```

```
{
    display:none;
}
</style>
</head>
<body>
    <span>内联元素</span>
    <div>块元素</div>
    <p>内联块元素</p>
    <img src="pic.jpg" alt="pic" />
</body>
</html>
```

在以上的代码中，定义了 4 个不同显示方式的元素，并分别定义了不同的属性值。<span>元素默认是内联元素，现定义为块元素。<div>元素默认是块元素，现定义为内联元素。<p>元素默认是块元素，现定义为列表元素。<img>元素默认为内联元素，现定义为不显示元素。代码运行后，其显示效果如图 18-15 所示。

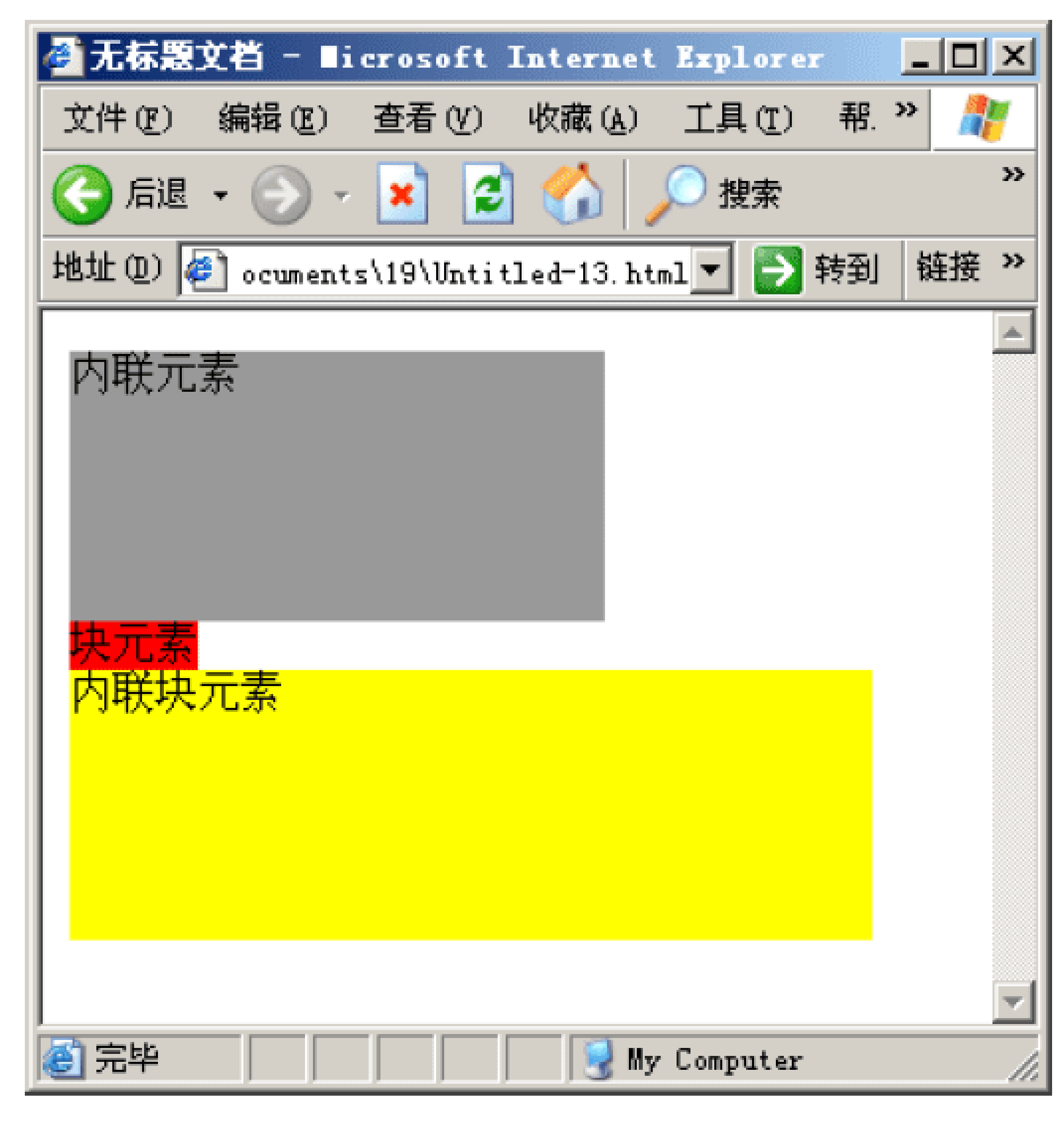


图 18-15 显示方式属性的显示效果

从图 18-15 可以看出，使用显示方式属性，可以方便地更改元素固有的显示方式。table-column-group 属性值和 table-footer-group 属性值，需要跨页显示，所以未在实例中使用。

18.5.2 内联块属性值的异常显示

在使用内联块属性值(inline-block)的时候，IE 浏览器对该属性值支持得并不完整。当在内联元素中使用时，元素可以显示块属性，同时保留原有的内联属性。当在块元素中使用时，元素可以显示块属性，但无法显示内联属性。

下面是一个使用内联块属性的实例，其代码如下所示。

例程 18-14　inline-block.html

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>无标题文档</title>
<style>
.span
{
display:inline-block;
width:200px;
height:100px;
background:blue;
}
span
{
background:red;
}
div
{
display:inline-block;
width:300px;
height:100px;
background:green;
}
</style>
</head>
<body>
<span class="span">定义内联块属性的内联元素</span>
<span>一个内联元素</span>
<div>定义内联块属性的块元素</div>
</body>
</html>
```

在以上的代码中，定义了 3 个元素。在第 1 个<span>元素中，定义元素的显示方式属性值为内联块属性。在第 2 个<span>元素中，只定义了元素的背景，目的是与其他元素作为对照。在<div>元素中，同样定义显示方式属性值为内联块属性。代码运行后，其显示效果如图 18-16 所示。

从图 18-16 可以看出，当在块元素中使用内联块属性值时，并不能改变块元素的显示方式。

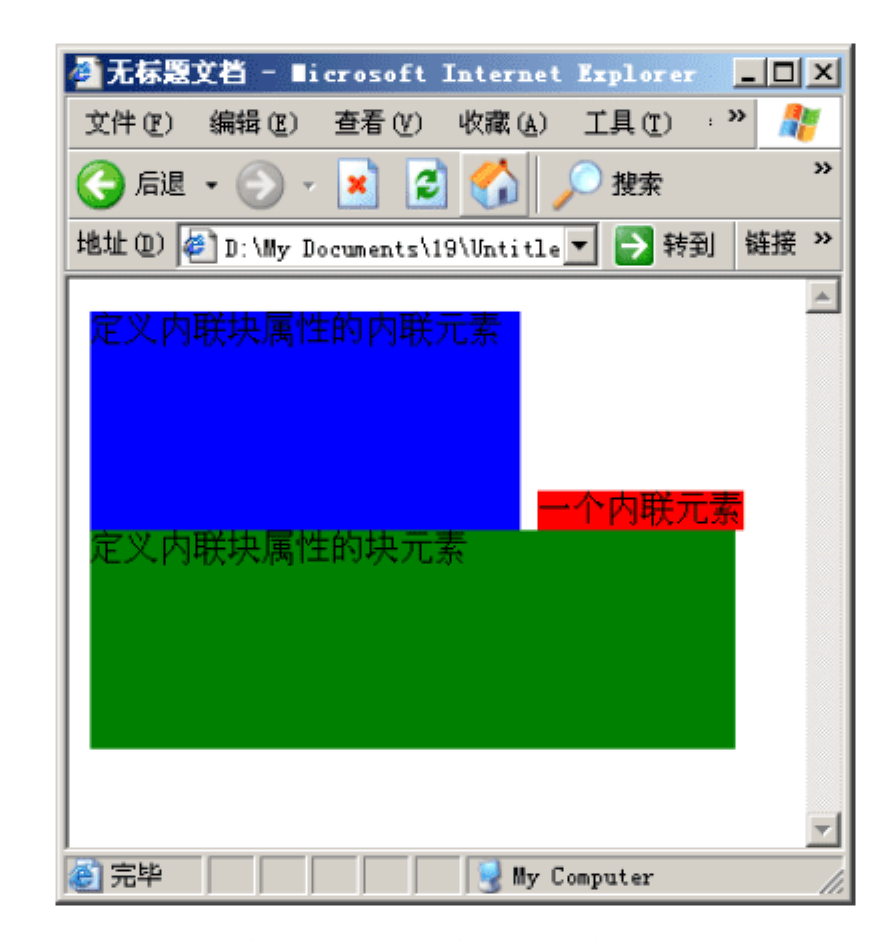


图 18-16　绝对定位属性局限性的显示效果 1

18.5.3　隐藏属性值 none

在使用隐藏属性值(none)的时候，不但隐藏了元

素包含的内容，同时元素所占有的物理空间也一起消失了。在页面中，其他元素将忽略隐藏元素的存在，按照各自原有的方式显示。

下面是一个使用隐藏属性值的实例，其代码如下所示。

例程 18-15 none.html

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>无标题文档</title>
<style>
span
{
background:red;
}
p
{
display:none;
width:200px;
height:100px;
background:green;
}
div
{
width:300px;
height:100px;
background:#666666;
}
</style>
</head>
<body>
<span>这是一个内联元素</span>
<p>这是一个使用 none 属性值的隐藏元素</p>
<div>这是一块元素</div>
</body>
</html>
```

在以上的代码中，定义了 3 个不同属性的元素。在<span>元素中，定义了元素的背景颜色。在<p>元素中，定义元素的显示属性值为 none，同时定义了元素的宽度、高度等属性。在<div>元素中，定义了元素的宽度、高度、背景等属性。定义<span>元素和<div>元素的目的是用来显示<p>元素中定义的属性值。代码运行后，其显示效果如图 18-17 所示。

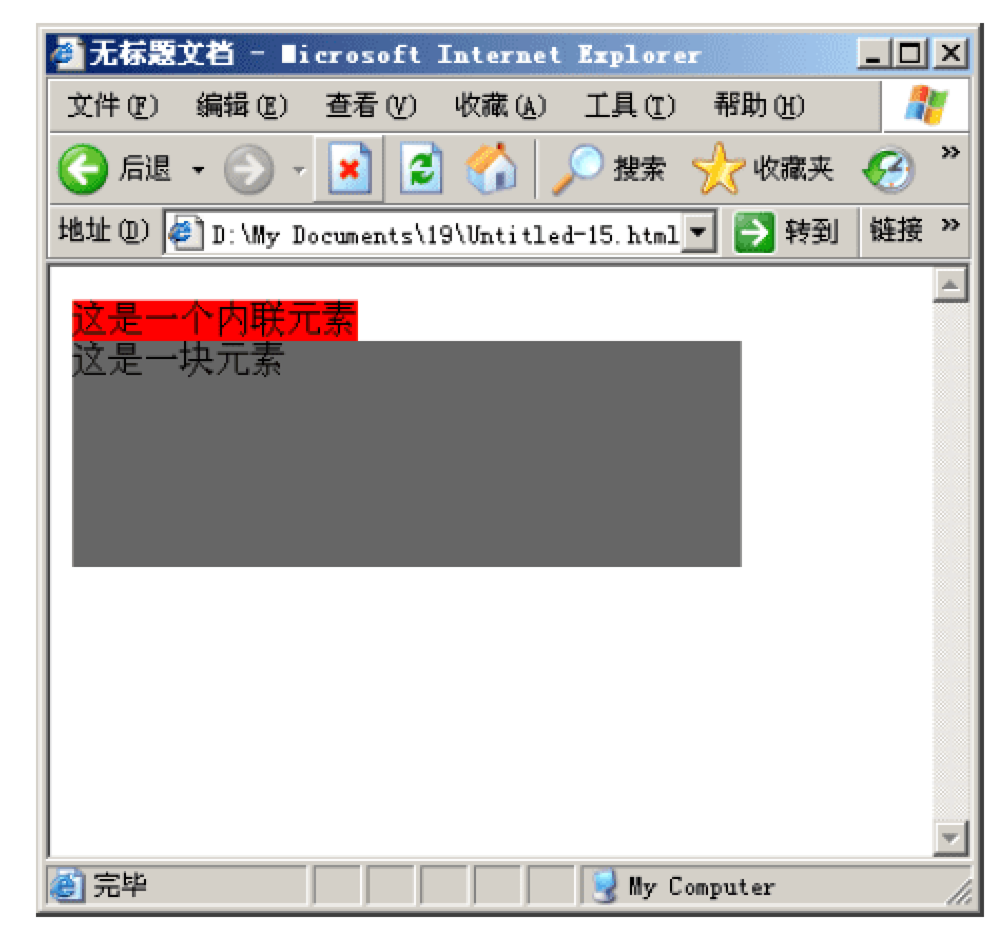


图 18-17　隐藏属性值的显示效果

从图 18-17 可以看出，由于使用了隐藏属性值，所以<span>元素和<div>元素会忽略<p>元素的存在。

18.6 元素的可视性

元素的可视性，是指元素是否可见。在 CSS 中，可以通过可视属性(visibility)，控制元素的可视性。使用可视属性，只能完全显示或者隐藏相应元素，无法显示部分元素内容。下面进行详细讲解。

18.6.1 可视属性 visibility

可视属性(visibility)用来定义元素及其内容是否可见。在显示方式属性中，可以使用 3 个属性值，分别为 visible、collapse 和 hidden，其语法结构如下所示。

```
visibility : visible | collapse | hidden;
```

其中部分属性值的含义如下所示。

- visible：定义元素正常显示。
- collapse：隐藏表格的行或列(浏览器未支持)。
- hidden：定义元素不显示。

下面是一个可视属性的实例，其代码如下所示。

例程 18-16　visibility.html

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>无标题文档</title>
```

```
05  <style>
06  div
    {
        visibility:hidden;
        width:200px;
        height:100px;
        background:yellow;
    }
07  span
    {
        background:red;
    }
08  </style>
09  </head>
10  <body>
    <div>这是隐藏可视性的元素</div>
    <span>这是一个内联元素</span>
11  </body></html>
```

以上的代码中，在<div>元素中定义了可视性属性值为隐藏，同时定义了元素的大小、背景等属性。在<span>元素中，定义了元素的背景为红色。此时，由于<div>元素中使用了隐藏属性值，所以<div>元素将无法显示。页面的显示效果如图 18-18 所示。

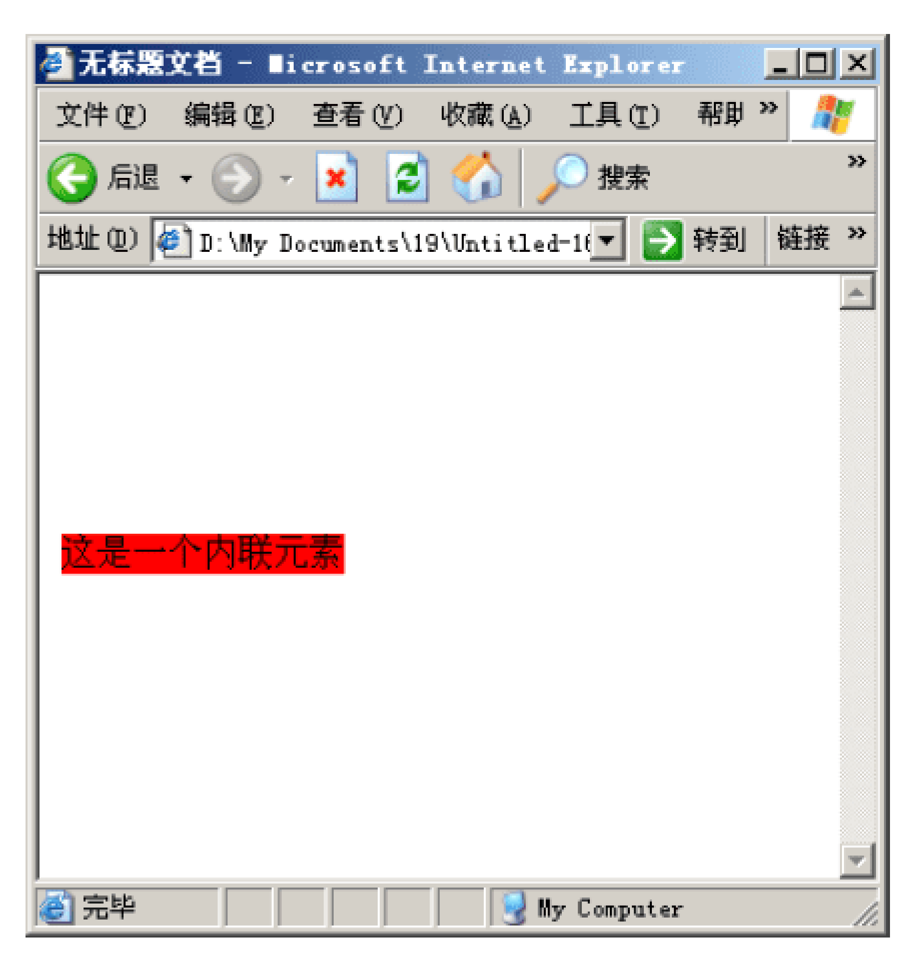


图 18-18　可视性属性的显示效果

从图 18-18 可以看出，使用可视性属性隐藏元素之后，元素占有的物理空间并没有发生变化。页面中的其他元素，按照隐藏元素依然存在的方式进行排列。

18.6.2　可视性属性与显示方式属性的关系

在 CSS 中，可视性属性用来控制元素是否显示，显示方式属性用来定义元素的显示方式。所以如果在显示方式属性中，定义了元素显示方式为隐藏(none)，则无论使用任何可视性属性值，元素依然无法显示。

下面是一个可视属性与显示方式属性的实例，其代码如下所示。

例程 18-17　example.html

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>无标题文档</title>
<style>
div
{
    display:none;
    visibility:visible;
    width:350px;
    height:150px;
    background:yellow;
}
P
{
  width:450px;
  height:150px;
  border:4px solid #000000;
  background:#999999;
}
</style>
</head>
<body>
<div>这是一个块元素</div>
<p>这是另一个块元素</p>
</body>
</html>
```

以上的代码中，在<div>元素中定义了可视性属性值为显示，同时定义元素显示方式为不显示。在<p>元素中，定义了元素的大小、边框、背景等属性，用来显示<div>元素是否依然占有物理空间。页面的显示效果如图 18-19 所示。

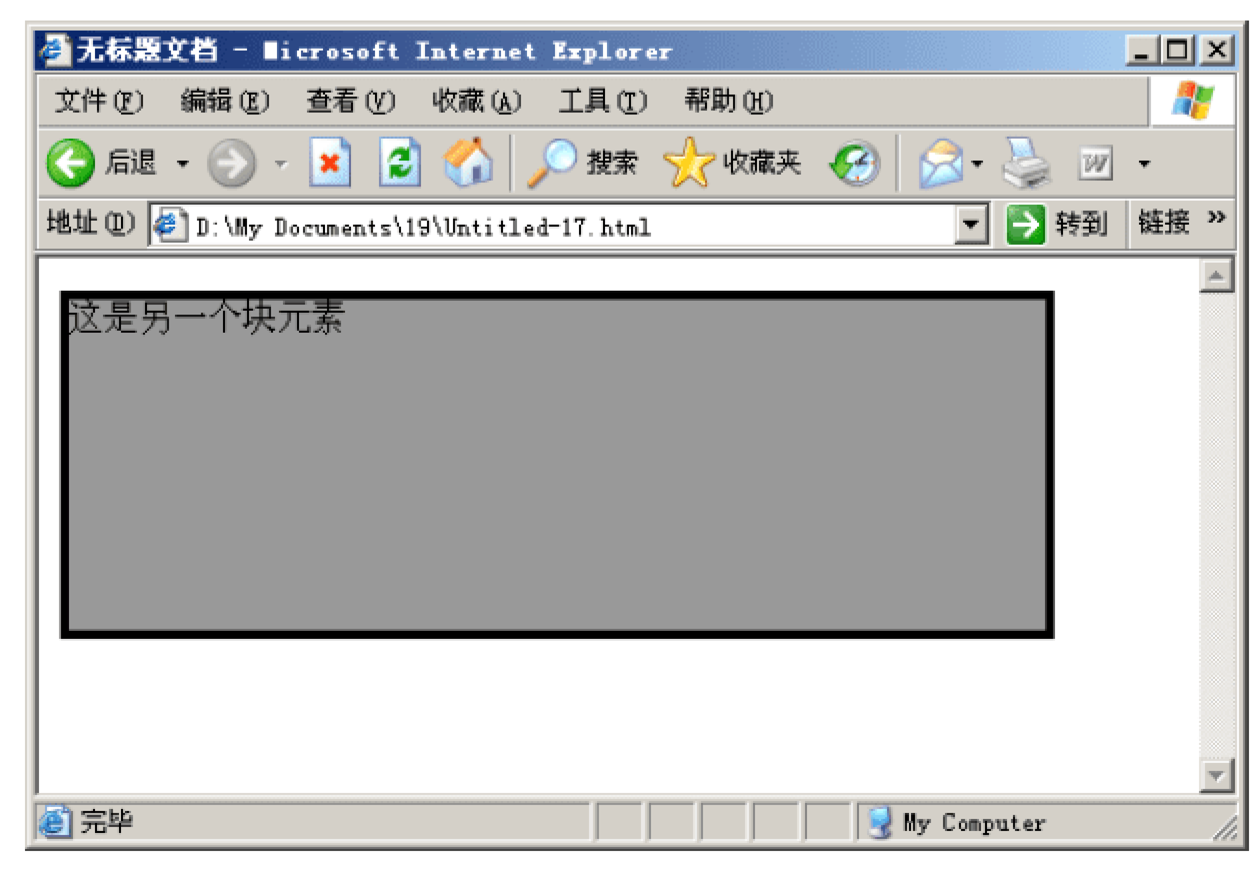


图 18-19　可视属性与显示方式属性的显示效果

从图 18-19 可以看出，由于在显示方式属性中，定义了隐藏属性值，所以即使在可视性属性中定义属性值为可见，依然无法显示元素及其内容。

18.7 本章习题

一、选择题

1．以下有关内容的剪切属性 clip 的说法中，错误的是：(　　)。

A. 在内容的剪切属性，可以使用两种属性值，一种为 auto 值，另一种为区域值

B. 内容的剪切属性(clip)，只能使用在绝对定位的元素中

C. 内容的剪切属性(clip)，也能使用在相对定位的元素中

D. 内容的剪切属性(clip)中 rect 中定义的 4 个属性值，是以元素左上角为中心，按照上、右、下、左的顺序定义剪切区域的四条边线，在四条边线划定的区域将显示元素内容

2．以下关于溢出属性(overflow)的说法中，错误的是：(　　)。

A. 使用 visible 属性，溢出内容按照原有的方式显示。

B. 使用 auto 属性当内容超出元素定义的大小的时候，显示滚条，否则正常显示内容。

C. 使用 hidden 属性会隐藏所有的内容。

D. 使用 scroll 属性总是会显示滚条。

3．display: list-item;表示的意思是：(　　)。

A. 定义元素按照块元素的方式显示

B. 定义元素隐藏

C. 定义元素按照内联元素的方式显示

D. 定义元素以列表元素的方式显示

4．清除浮动属性 clear 的属性值不包括(　　)。

A. left　　B. top　　C. right　　D. bottom

二、填空题

1．元素的浮动属性(float)用来______________。可以使用 4 种属性值，分别为_______、______、________、_________。

2．内容的剪切属性(clip)用来__________。可以使用两种属性值，一种为_______值，另一种为区域值

3．溢出属性(overflow)用来定义____________。在溢出属性中，可以使用 4 个属性值，分别为_______、________、_______、________。

4．在 CSS 中，更改元素显示方式使用的是显示方式属性是________。

三、实战练习

1．制作页面，同时使用 4 个浮动元素.

2．制作页面，练习使用溢出属性(overflow)的各个属性值。

3．制作页面，练习使用可视性属性。注意可视性属性与显示方式属性的关系。

Chapter 19

CSS控制其他元素的显示

CSS 控制其他元素的显示，包括很多方面的内容，例如滚条、光标、打印等。其中有部分内容要通过 CSS 中的伪类选择符实现。在 CSS 中，关于其他元素的修饰属性有很多，但大多数还不被浏览器所支持。所以本章只讲解其中被浏览器所支持的各种属性和伪类。

本章主要内容有:

◎ 熟悉 CSS 控制滚条的显示的方法。

◎ 了解 CSS 控制光标的显示的方法。

◎ 学会控制元素的缩放。

◎ 精通控制超链接的显示效果。

19.1 控制滚条的显示

在各种浏览器中都存在滚条，目的是在浏览器所在的空间中能够包含更多的内容。可以通过定义 CSS 样式，显示元素的滚条。通过使用滚条的相关属性，可以修饰和美化页面中各个位置的滚条。具体内容如下所示。

注意

滚条的相关属性是 IE 浏览器的特有属性，在其他浏览器中无法显示滚条属性中定义的显示效果。

19.1.1 滚条 3d 亮边框颜色属性 scrollbar-3dlight-color

一般页面中滚条都显示在浏览器的右侧。滚条 3d 亮边框，是指右侧滚条中显示在滑块和箭头区域最外层左侧和顶部的区域。滚条 3d 亮边框颜色属性的语法结构如下所示。

```
scrollbar-3dlight-color : color;
```

下面是一个使用滚条 3d 亮边框颜色属性的实例，其代码如下所示。

例程 19-1 scrollbar-3dlight-color.html

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>无标题文档</title>
<style>
html
  {
    scrollbar-3dlight-color:red;
  }
div
  {
   height:100px;
   width:300px;
   padding:100px 0 0 0;
   border:4px solid #000000;
   overflow:scroll;
  }
</style>
</head>
<body>
<div>注意滚条边框的颜色<br />注意滚条边框的颜色<br />注意滚条边框的颜色<br />注意滚条边框的颜色色<br />注意滚条边框的颜色<br /></div>
```

```
12    </body>
13    </html>
```

以上的代码中，06 行在<html>元素中，定义了滚条 3d 亮边框颜色属性值为红色。同时定义了<html>元素中的<div>元素的宽度、高度、边框等属性，并定义其滚条显示效果为始终显示。在<div>元素中，定义了文本内容，目的是使<div>元素中产生滚条。代码运行后，可以看到滚条的边框为红色，页面的显示效果如图 19-1 所示。

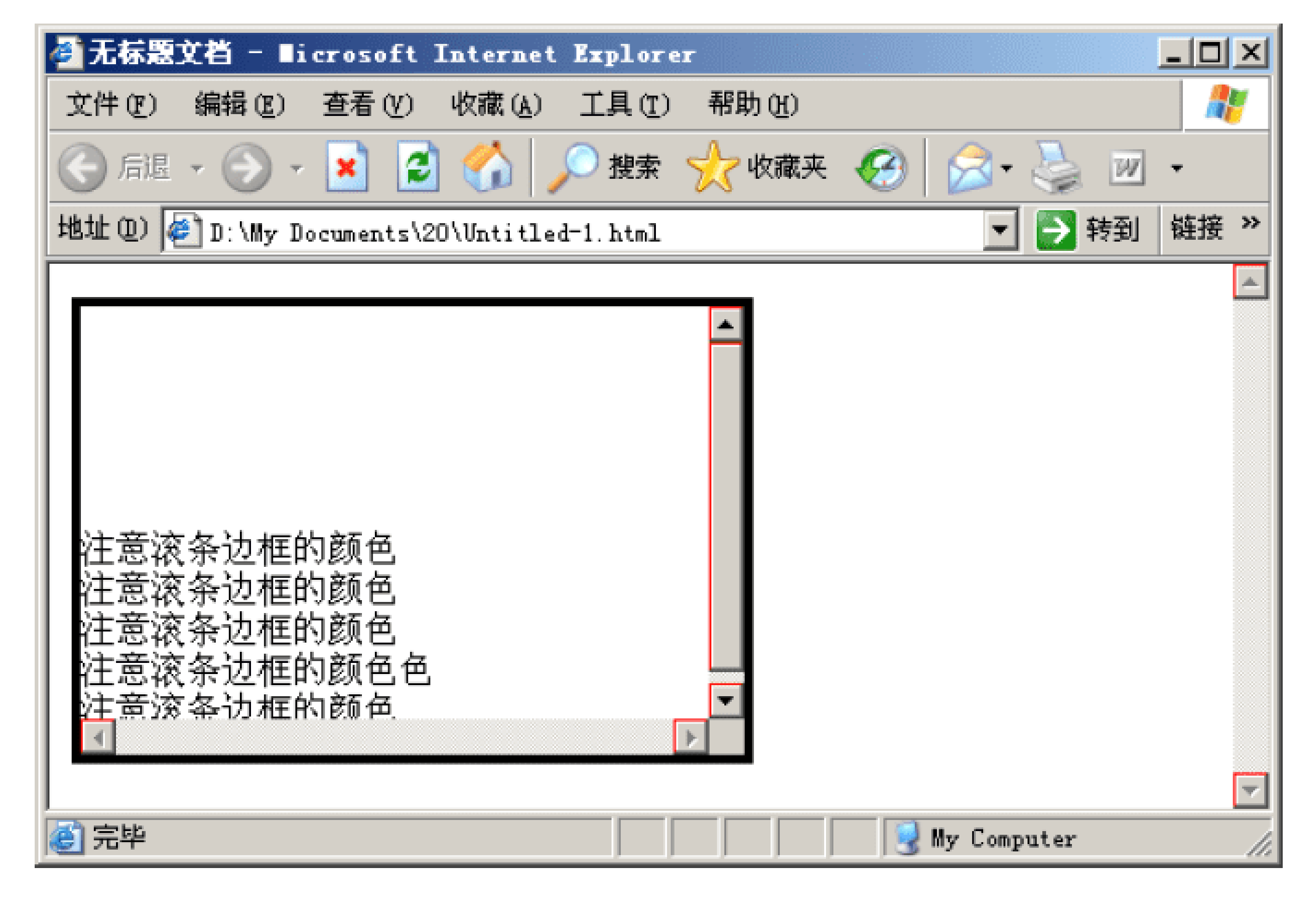


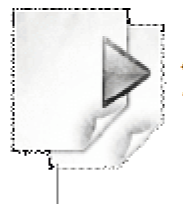
图 19-1　滚条 3d 亮边框颜色属性的显示效果

从图 19-1 可以看到，当在<html>元素中定义了滚条颜色属性后，其子元素中都将继承这个属性中定义的显示效果。

19.1.2　滚条亮边框颜色属性 scrollbar-highlight-color

滚条亮边框，是指右侧显示的滚条中，与滚条 3d 亮边框相邻的区域。滚条亮边框颜色属性的语法结构如下所示。

```
scrollbar-highlight-color : color;
```



注意

在单独使用滚条亮边框颜色属性时，将对滚条其他部分的颜色产生影响。

下面是一个使用滚条亮边框颜色属性的实例，其代码如下所示。

例程 19-2　scrollbar-highlight-color.html

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>无标题文档</title>
<style>
```

```
06  html
    {
        scrollbar-highlight-color:blue;
    }
07  div
    {
        height:80px;
        width:400px;
        padding:100px 0 0 0;
        border:4px solid #666666;
        overflow:scroll;
    }
08  </style>
09  </head>
10  <body>
11  <div>滚条亮边框颜色属性<br />滚条亮边框颜色属性<br />滚条亮边框颜色属性<br />滚条亮边框颜色属性<br />滚条亮边框颜色属性<br /></div>
12  </body>
13  </html>
```

以上的代码中，06 行在<html>元素中，定义了滚条亮边框颜色属性值为黑色。同时 07 行定义了<html>元素中的<div>元素的宽度、高度、边框等属性，并定义其滚条显示效果为始终显示。在<div>元素中，定义了文本内容，目的是使<div>元素中产生滚条。代码运行后，当拖动浏览器的窗口高度小于内容时，页面的显示效果如图 19-2 所示。

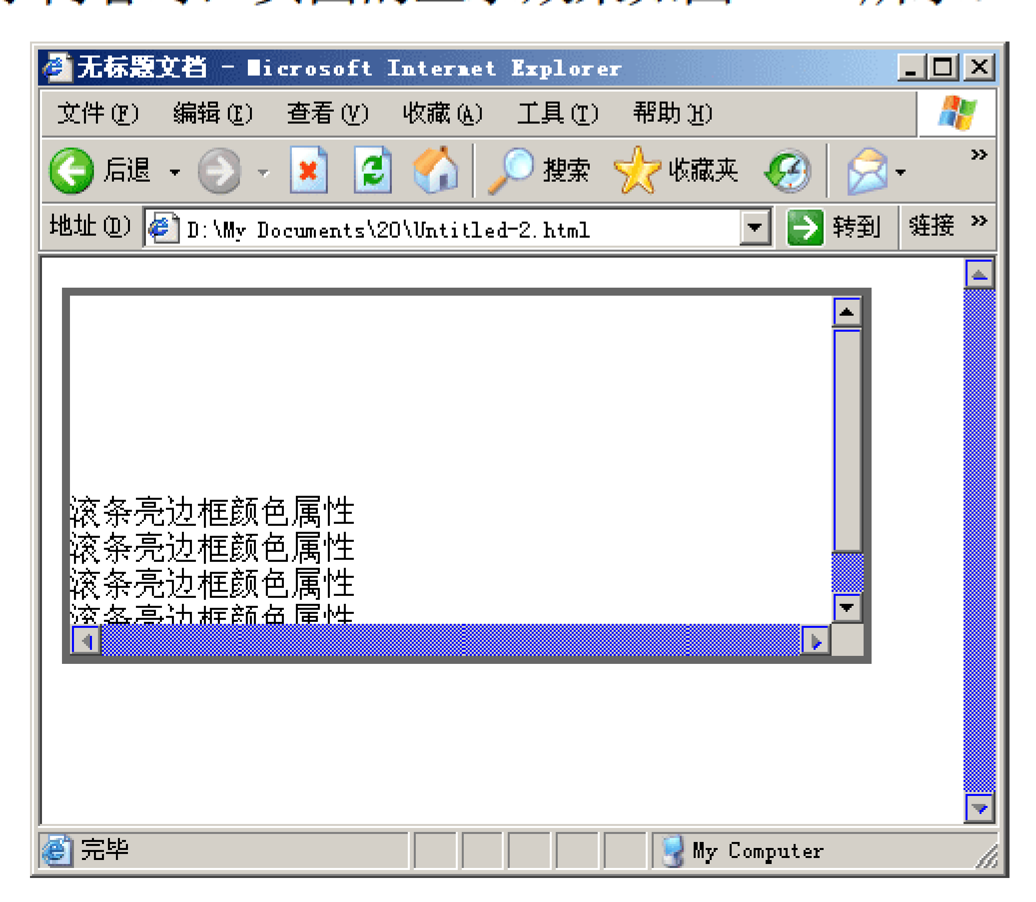


图 19-2　滚条亮边框颜色属性的显示效果

19.1.3　滚条滑块颜色属性 scrollbar-face-color

滚条滑块，是指右侧显示的滚条中，可以使用鼠标上下拖动的区域，以及箭头区域部分。滚条滑块颜色属性的语法结构如下所示。

```
scrollbar-face-color : color;
```

下面是一个使用滚条滑块颜色属性的实例，其代码如下所示。

例程 19-3 scrollbar-face-color.html

```
01 <html xmlns="http://www.w3.org/1999/xhtml">
02 <head>
03 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
04 <title>无标题文档</title>
05 <style>
06 html
   {
     scrollbar-face-color:blue;
   }
07 div
   {
     height:100px;
     width:400px;
     padding:100px 0 0 0;
     border:1px solid #666666;
     overflow:scroll;
   }
08 </style>
09 </head>
10 <body>
11 <div>滑块的颜色<br />滑块的颜色<br />滑块的颜色<br />滑块的颜色<br />滑块的颜色<br /></div>
12 </body>
13 </html>
```

以上的代码中，06 行在<html>元素中，定义了滚条滑块颜色属性值为黑色。同时定义了<html>元素中的<div>元素的宽度、高度、边框等属性，并定义其滚条显示效果为始终显示。在<div>元素中，定义了文本内容，目的是使<div>元素中产生滚条。代码运行后，当拖动浏览器的窗口高度小于内容时，页面的显示效果如图 19-3 所示。

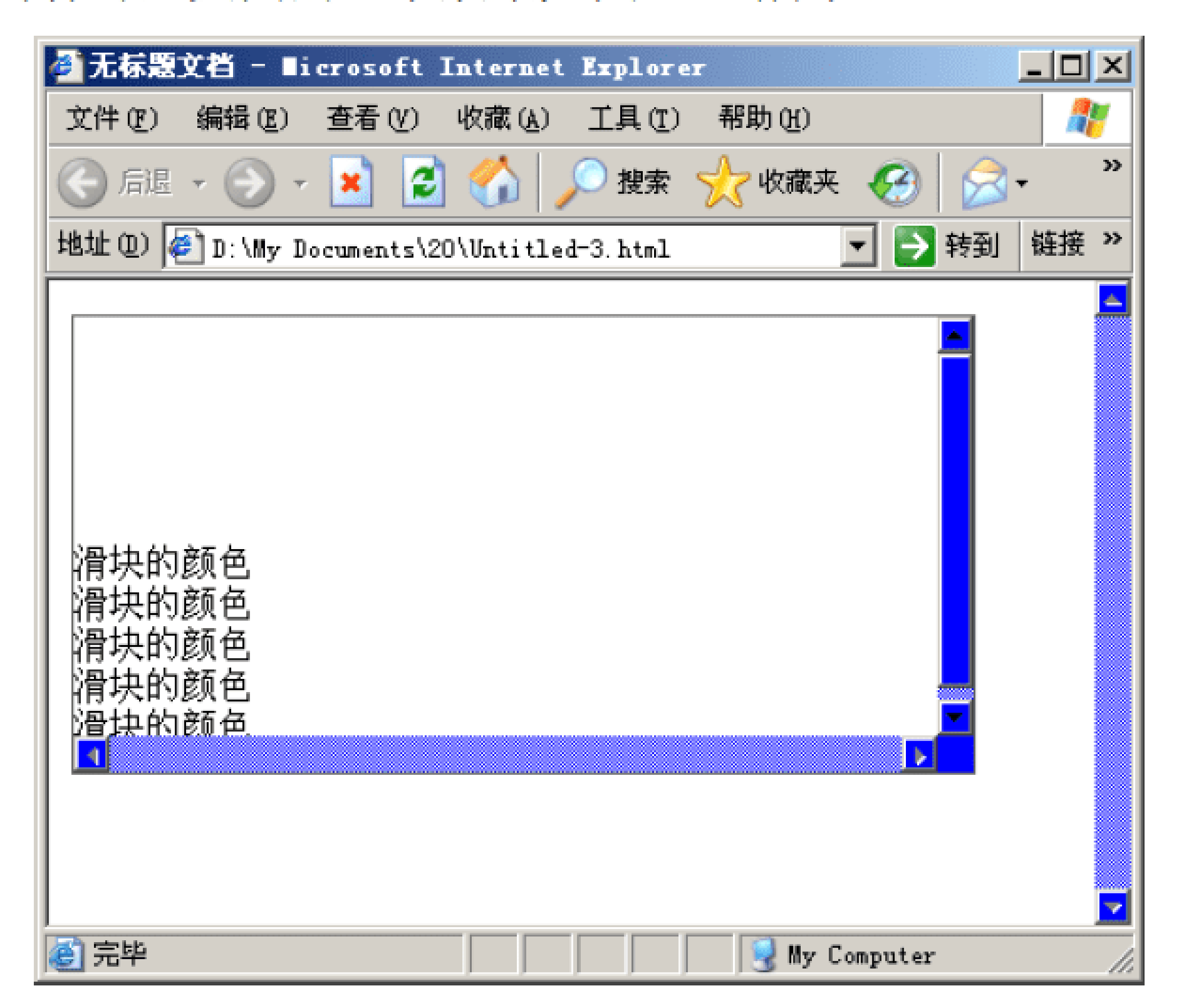


图 19-3 滚条滑块颜色属性的显示效果

19.1.4 滚条箭头颜色属性 scrollbar-arrow-color

滚条箭头，是指右侧滚条中，在滚条的顶部和底部显示的三角箭头的区域。滚条箭头颜色属性的语法结构如下所示。

```
scrollbar-arrow-color : color;
```

下面是一个使用滚条箭头颜色属性的实例，其代码如下所示。

例程 19-4 scrollbar-arrow-color.html

```
01  <html xmlns="http://www.w3.org/1999/xhtml">
02  <head>
03  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
04  <title>无标题文档</title>
05  <style>
06  div
    {
    scrollbar-arrow-color:block;
    height:100px;
    width:400px;
    padding:100px 0 0 0;
    border:1px solid #666666;
    overflow:scroll;
  }
07  </style>
08  </head>
09  <body>
10  <div>注意滚条的箭头颜色<br />注意滚条的箭头颜色<br />注意滚条的箭头颜色<br />注意滚条的箭头颜色<br />注意滚条的箭头颜色<br /></div>
11  </body>
12  </html>
```

以上的代码中，06 行中在<div>元素中定义了滚条箭头的颜色为白色。同时定义了<div>元素的宽度、高度、边框等属性，并定义其滚条显示效果为始终显示。在<div>元素中，定义了文本内容，目的是使<div>元素中产生滚条。代码运行后，当拖动浏览器的窗口高度小于内容时，页面的显示效果如图 19-4 所示。

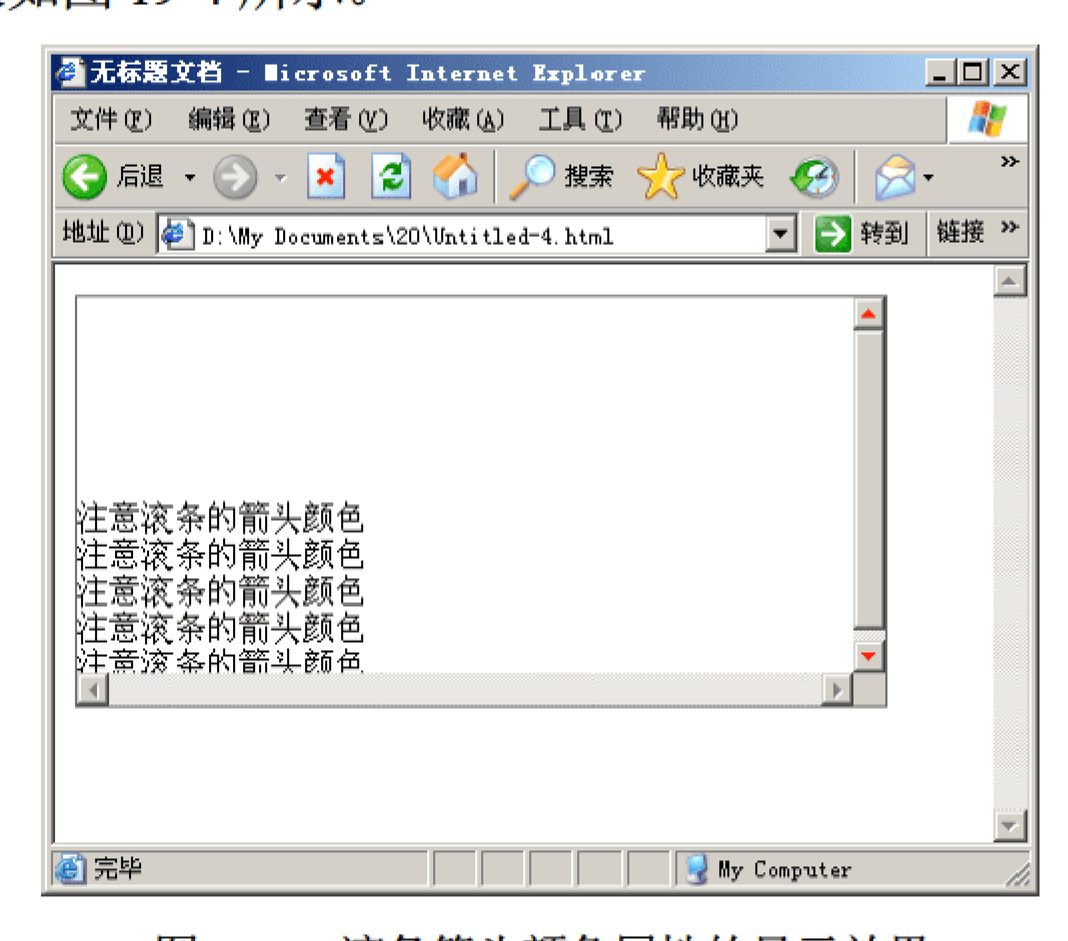


图 19-4 滚条箭头颜色属性的显示效果

19.1.5 滚条阴影颜色属性 scrollbar-shadow-color

滚条阴影，是指右侧滚条中，显示在滚条拖动区域和箭头区域右侧和底部的部分。滚条阴影颜色属性的语法结构如下所示。

```
scrollbar-shadow-color : color;
```

下面是一个使用滚条阴影颜色属性的实例，其代码如下所示。

例程 19-5 scrollbar-shadow-color.html

```
01  <html xmlns="http://www.w3.org/1999/xhtml">
02  <head>
03  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
04  <title>无标题文档</title>
05  <style>
06  div
    {
        scrollbar-shadow-color:blue;
        height:100px;
        width:400px;
        padding:100px 0 0 0;
        border:1px solid #666666;
        overflow:scroll;
    }
07  </style>
08  </head>
09  <body>
10  <div>这里是元素内容部分<br />这里是元素内容部分<br />这里是元素内容部分<br />这里是元素内容部分<br />这里是元素内容部分<br /></div>
11  </body>
12  </html>
```

以上的代码中，06 行中在<div>元素中，定义滚条阴影的颜色为黑色。同时定义了<div>元素的宽度、高度、边框等属性，并定义其滚条显示效果为始终显示。在<div>元素中，定义了文本内容，目的是使<div>元素中产生滚条。代码运行后，当拖动浏览器的窗口高度小于内容时，页面的显示效果如图 19-5 所示。

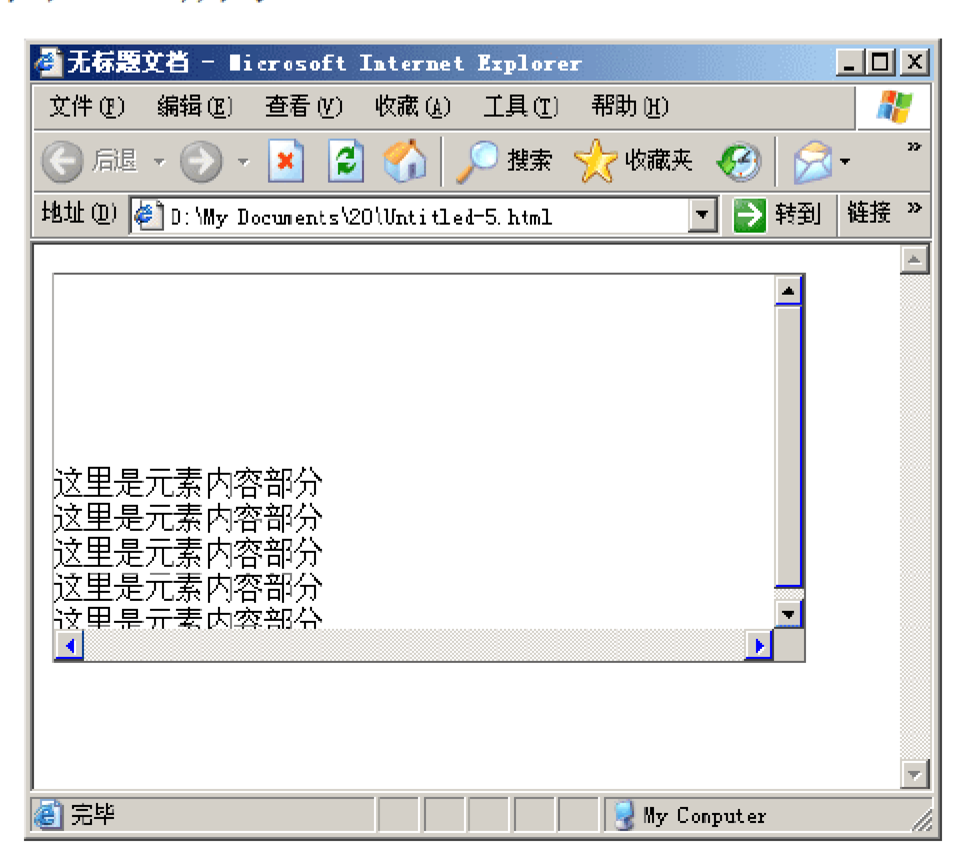


图 19-5 滚条阴影颜色属性的显示效果

19.1.6 滚条暗部阴影属性 scrollbar-darkshadow-color

滚条暗部阴影，是指右侧滚条中，显示在与滚条阴影相邻的左侧和顶部区域。滚条暗部阴影颜色属性的语法结构如下所示。

```
scrollbar-darkshadow-color: color;
```

下面是一个使用滚条暗部阴影颜色属性的实例，其代码如下所示。

例程 19-6 scrollbar-darkshadow-color.html

```
01  <html xmlns="http://www.w3.org/1999/xhtml">
02  <head>
03  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
04  <title>无标题文档</title>
05  <style>
06  div
  {
    scrollbar-darkshadow-color:yellow;
    height:100px;
    width:400px;
    padding:100px 0 0 0;
    border:1px solid #666666;
    overflow:scroll;
  }
07  </style>
08  </head>
09  <body>
10  <div>how are you <br />how are you <br />how are you <br />how are you <br />how are you <br /></div>
11  </body>
12  </html>
```

以上的代码中，06 行中在<div>元素定义了滚条暗部阴影的颜色为黄色。同时定义了<div>元素的宽度、高度、边框等属性，并定义其滚条显示效果为始终显示。在<div>元素中，定义了文本内容，目的是使<div>元素中产生滚条。代码运行后，当拖动浏览器的窗口高度小于内容时，页面的显示效果如图 19-6 所示。

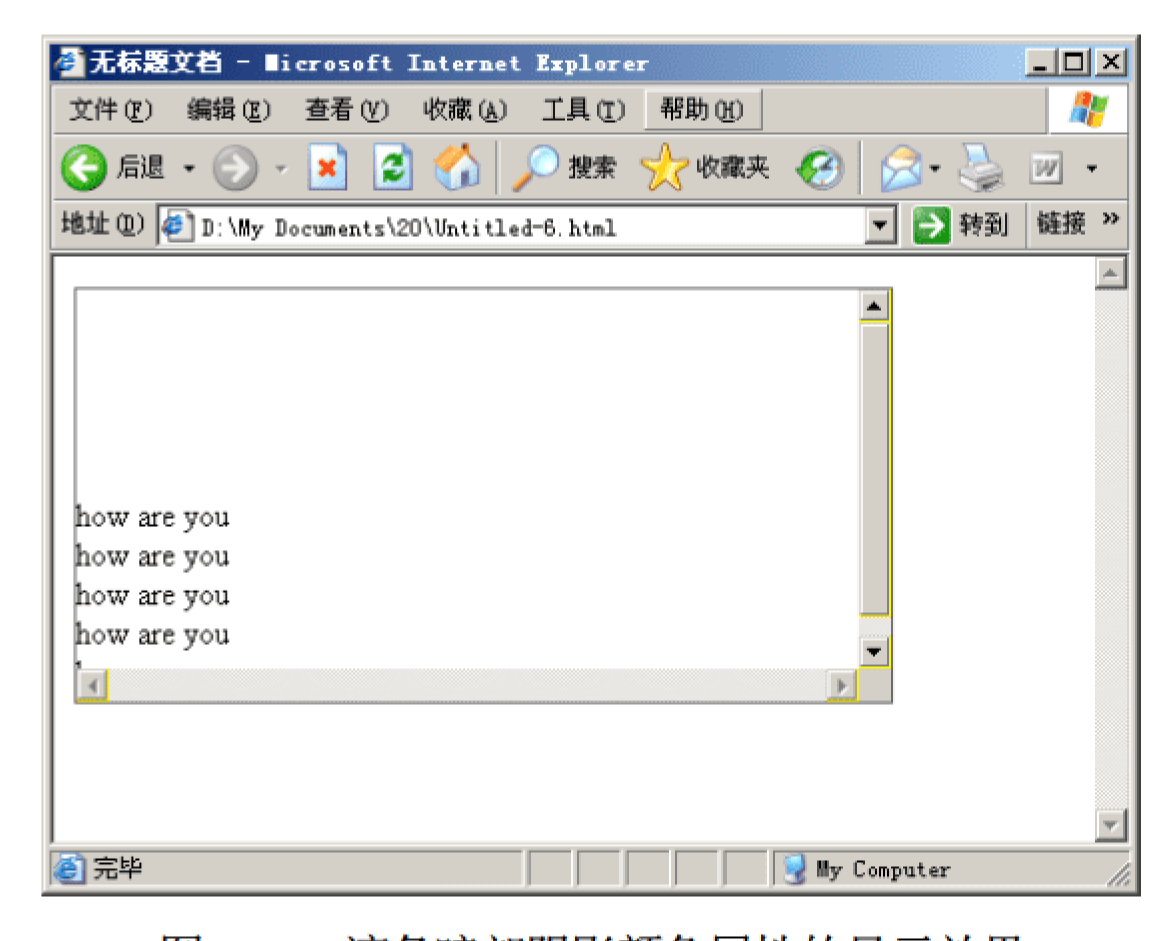


图 19-6 滚条暗部阴影颜色属性的显示效果

19.1.7 滚条拖动区颜色属性 scrollbar-track-color

滚条拖动区，是指右侧滚条中，可以使用鼠标进行上下拖动的区域的背景部分。滚条拖动区颜色属性的语法结构如下所示。

```
scrollbar-track-color : color;
```

下面是一个使用滚条拖动区颜色属性的实例，其代码如下所示。

例程 19-7 scrollbar-track-color.html

```
01  <html xmlns="http://www.w3.org/1999/xhtml">
02  <head>
03  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
04  <title>无标题文档</title>
05  <style>
06  div
    {
    scrollbar-track-color:red;
    height:100px;
    width:300px;
    padding:100px 0 0 0;
    border:4px solid #666666;
    overflow:scroll;
  }
07  </style>
08  </head>
09  <body>
10  <div>how are you<br />how are you<br />how are you<br />how are you<br />how are you<br />how are
you<br />how are you<br />how are you<br /></div>
11  </body>
12  </html>
```

以上的代码中，06 行中在<div>元素中，定义了滚条拖动区的颜色属性值为红色。同时定义了<div>元素的宽度、高度、边框等属性，并定义其滚条显示效果为始终显示。在<div>元素中，定义了文本内容，目的是使<div>元素中产生滚条。代码运行后，当拖动浏览器的窗口高度小于内容时，页面的显示效果如图 19-7 所示。

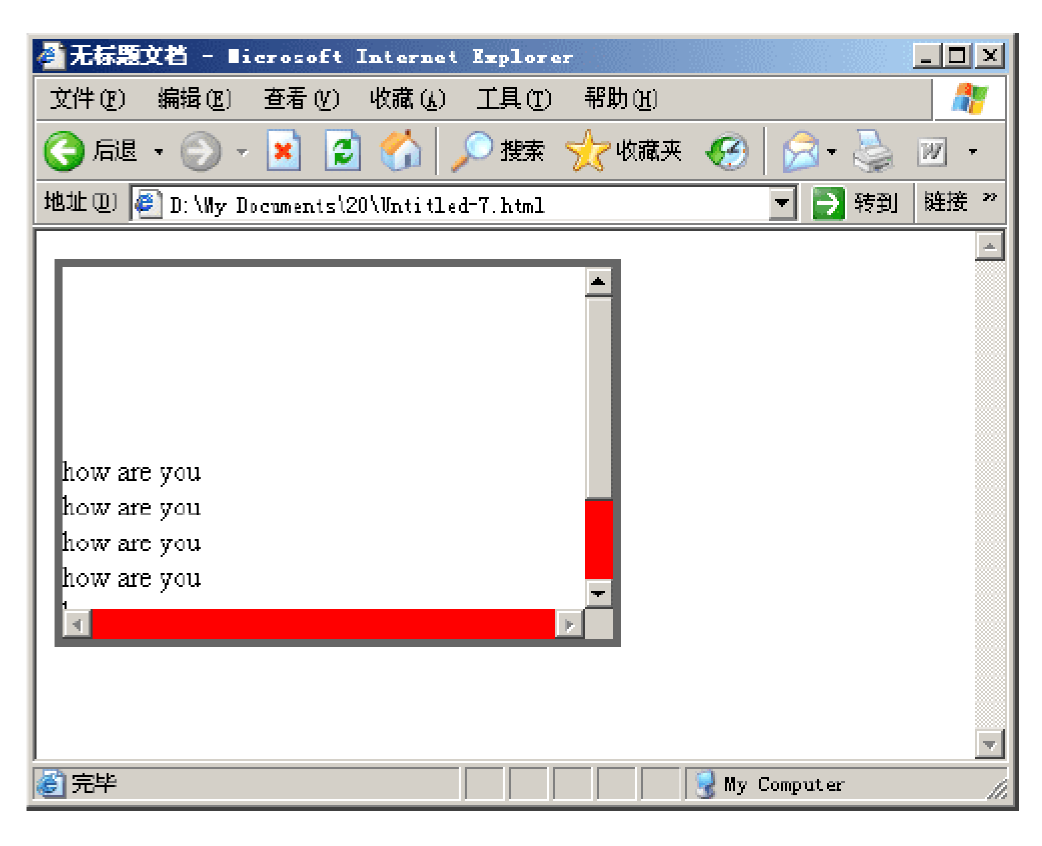


图 19-7 滚条拖动区颜色属性的显示效果

19.1.8 滚条基准色属性 scrollbar-base-color

滚条基准色，是指在滚条中定义一个作为基准的颜色，滚条所有部分按照基准颜色自动调整显示。滚条基准色属性的语法结构如下所示。

```
scrollbar-base-color : color;
```

下面是一个使用滚条基准色属性的实例，其代码如下所示。

例程 19-8 scrollbar-base-color.html

```
01  <html xmlns="http://www.w3.org/1999/xhtml">
02  <head>
03  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
04  <title>无标题文档</title>
05  <style>
06  div
    {
    scrollbar-base-color:blue;
    height:100px;
    width:300px;
    padding:100px 0 0 0;
    border:4px solid #666666;
    overflow:scroll;
    }
07  </style>
08  </head>
09  <body>
10  <div>how are you<br />how are you<br />how are you<br />how are you<br />how are you<br /></div>
11  </body>
12  </html>
```

以上的代码中，06 行中在<div>元素定义滚条的基准颜色为蓝色。同时定义了<div>元素的宽度、高度、边框等属性，并定义其滚条显示效果为始终显示。在<div>元素中，定义了文本内容，目的是使<div>元素中产生滚条。代码运行后，当拖动浏览器的窗口高度小于内容时，页面的显示效果如图 19-8 所示。

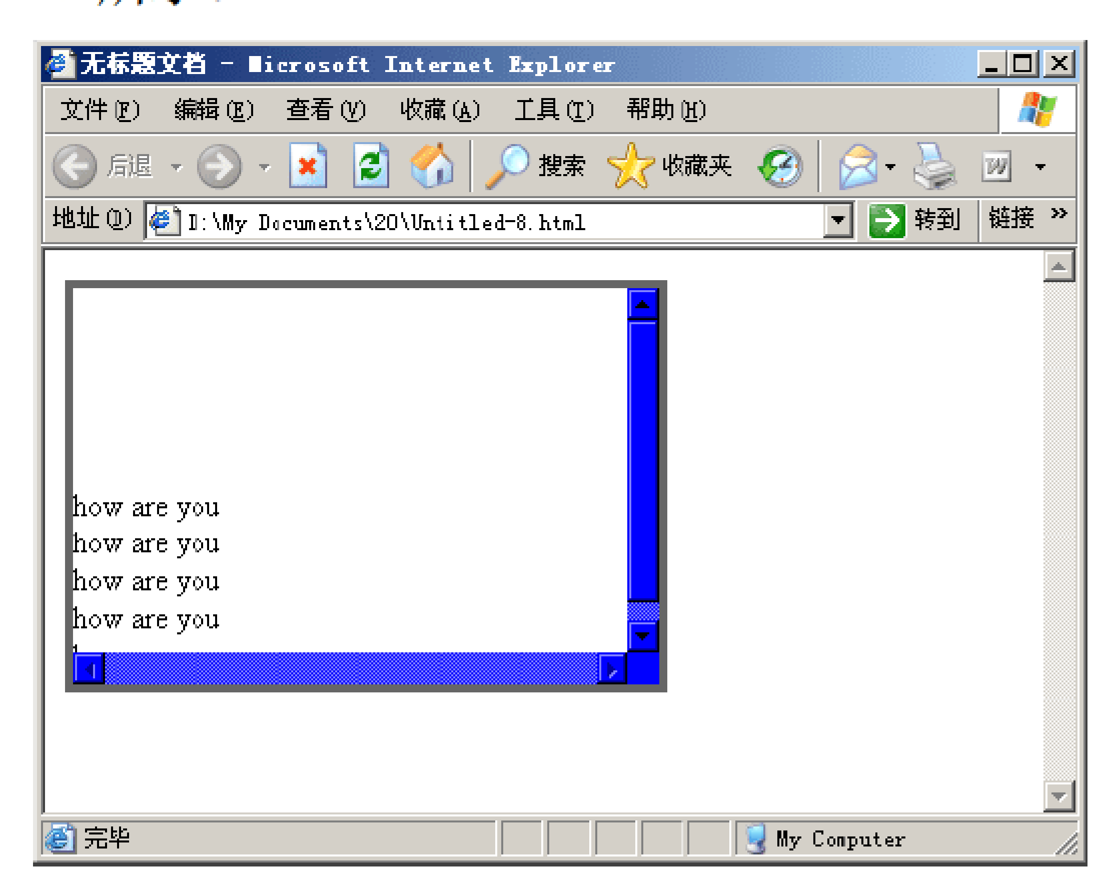


图 19-8 滚条基准色属性的显示效果

19.1.9 定义滚条的颜色

在定义滚条颜色的时候，可以同时使用除基准色属性之外的各种属性。因为如果定义了基准色属性，则基准色会和其他区域定义的颜色发生冲突。

下面是一个定义滚条颜色的实例，其代码如下所示。

例程 19-9 example.html

```
01  <html xmlns="http://www.w3.org/1999/xhtml">
02  <head>
03  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
04  <title>无标题文档</title>
05  <style>
06  div
  {
    scrollbar-3dlight-color:red;
    scrollbar-highlight-color:#000000;
    scrollbar-face-color:blue;
    scrollbar-arrow-color:#ffffff;
    scrollbar-shadow-color:#cccccc;
    scrollbar-darkshadow-color:#999999;
    scrollbar-track-color:#dddddd;
    height:100px;
    width:400px;
    padding:100px 0 0 0;
    border:1px solid #666666;
    overflow:scroll;
  }
07  </style>
08  </head>
09  <body>
10  <div>how are you<br />how are you<br />how are you<br />how are you<br />how are you<br /></div>
11  </body>
12  </html>
```

以上的代码中，06 行中在<div>元素中定义了滚条区域的各部分的颜色。同时定义了<div>元素的宽度、高度、边框等属性，并定义其滚条显示效果为始终显示。在<div>元素中，定义了文本内容，目的是使<div>元素中产生滚条。代码运行后，当拖动浏览器的窗口高度小于内容时，页面的显示效果如图 19-9 所示。

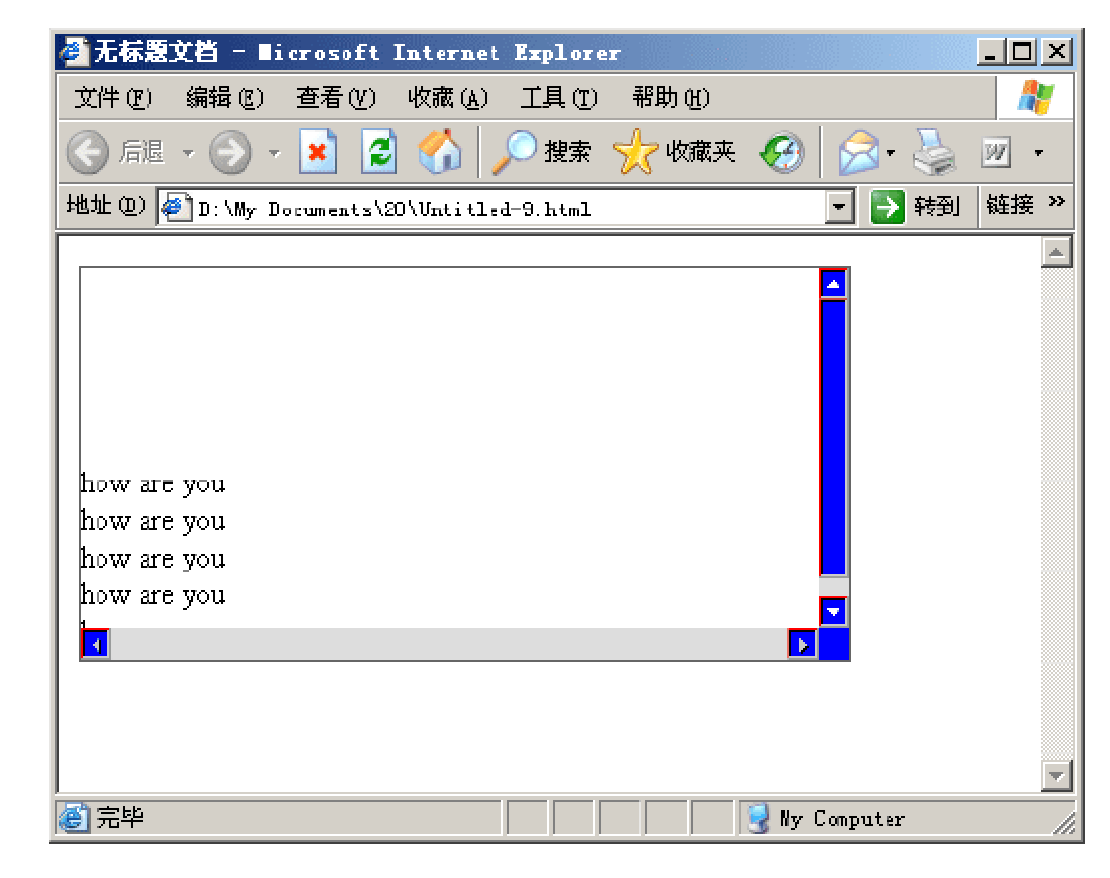


图 19-9 滚条颜色的显示效果

19.2 控制光标的显示

在 CSS 中，可以使用相应的属性，定义光标的显示效果。在网页中特定的内容部分会显示特定的光标效果，通过使用 CSS 中的光标控制属性，可以改变这种默认的显示方式，使显示效果更加合理。光标控制属性的语法结构如下所示。

```
cursor : auto | all-scroll | col-resize| crosshair | default | hand | move | help | no-drop | not-allowed | pointer | progress | row-resize | text | vertical-text | wait | *-resize | url ;
```

其中部分属性值的含义如下所示。

- all-scroll：使用中间圆点，上下左右四个箭头的显示效果。
- col-resize：使用中间竖线，左右两个箭头的显示效果。
- crosshair：使用十字线的显示效果。
- default：使用客户端平台的默认光标。通常是一个箭头。
- hand：使用手形的光标效果。
- move：使用十字箭头的显示效果。
- help：使用带问号标记的箭头的显示效果。
- no-drop：使用带有禁止标志的手形光标效果
- not-allowed：使用禁止标记的显示效果。
- pointer：使用手形的光标效果。
- progress：使用带有沙漏标记的箭头光标效果。
- row-resize：使用中间横线，上下两个箭头的显示效果。
- text：使用大写字母 I 的显示效果。
- vertical-text：使用大写字母 I 旋转 90 度的显示效果。
- wait：使用沙漏或手表的显示效果。
- *-resize：使用带有方向的箭头效果。分为 w-resize、s-resize、n-resize、e-resize、ne-resize、sw-resize、se-resize、nw-resize 等几个。
- url：使用用户自定义光标。通常后缀为.cur 或者.ani。

下面是一个使用光标控制属性的实例，其代码如下所示。

例程 19-10 example.html

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>css 属性实例</title>
<style>
.p1{
cursor:auto;
}
```

```
.p2{
  cursor:crosshair;
}
.p2{
  cursor:col-resize;
}
.p3{
  cursor:default;
}
.p4{
cursor:hand;
}
.p5{
cursor: move;
}
.p6{
  cursor:help;
}
.p7{
cursor:no-drop;
}
.p8{
  cursor:not-allowed;
}
.p9{
  cursor:pointer;
}
.p10{
  cursor:progress;
}
.p11{
  cursor:row-resize;
}
.p12{
cursor:text;
}
.p13{
cursor:vertical-text;
}
.p14{
cursor:wait;
}
p{
height:20px;
background:#999999;
border:1px solid #000000;
margin:0 0 10px 0;
}
</style>
</head>
```

```
<body>
<p class="p1">这里是光标效果属性的实例</p>
<p class="p2">这里是光标效果属性的实例</p>
<p class="p3">这里是光标效果属性的实例</p>
<p class="p4">这里是光标效果属性的实例</p>
<p class="p5">这里是光标效果属性的实例</p>
<p class="p6">这里是光标效果属性的实例</p>
<p class="p7">这里是光标效果属性的实例</p>
<p class="p8">这里是光标效果属性的实例</p>
<p class="p9">这里是光标效果属性的实例</p>
<p class="p10">这里是光标效果属性的实例</p>
<p class="p11">这里是光标效果属性的实例</p>
<p class="p12">这里是光标效果属性的实例</p>
<p class="p13">这里是光标效果属性的实例</p>
<p class="p14">这里是光标效果属性的实例</p>
</body>
</html>
```

以上的代码中，在每个<p>元素中，定义了不同的光标显示属性。代码运行后的显示效果如图 19-10 所示。

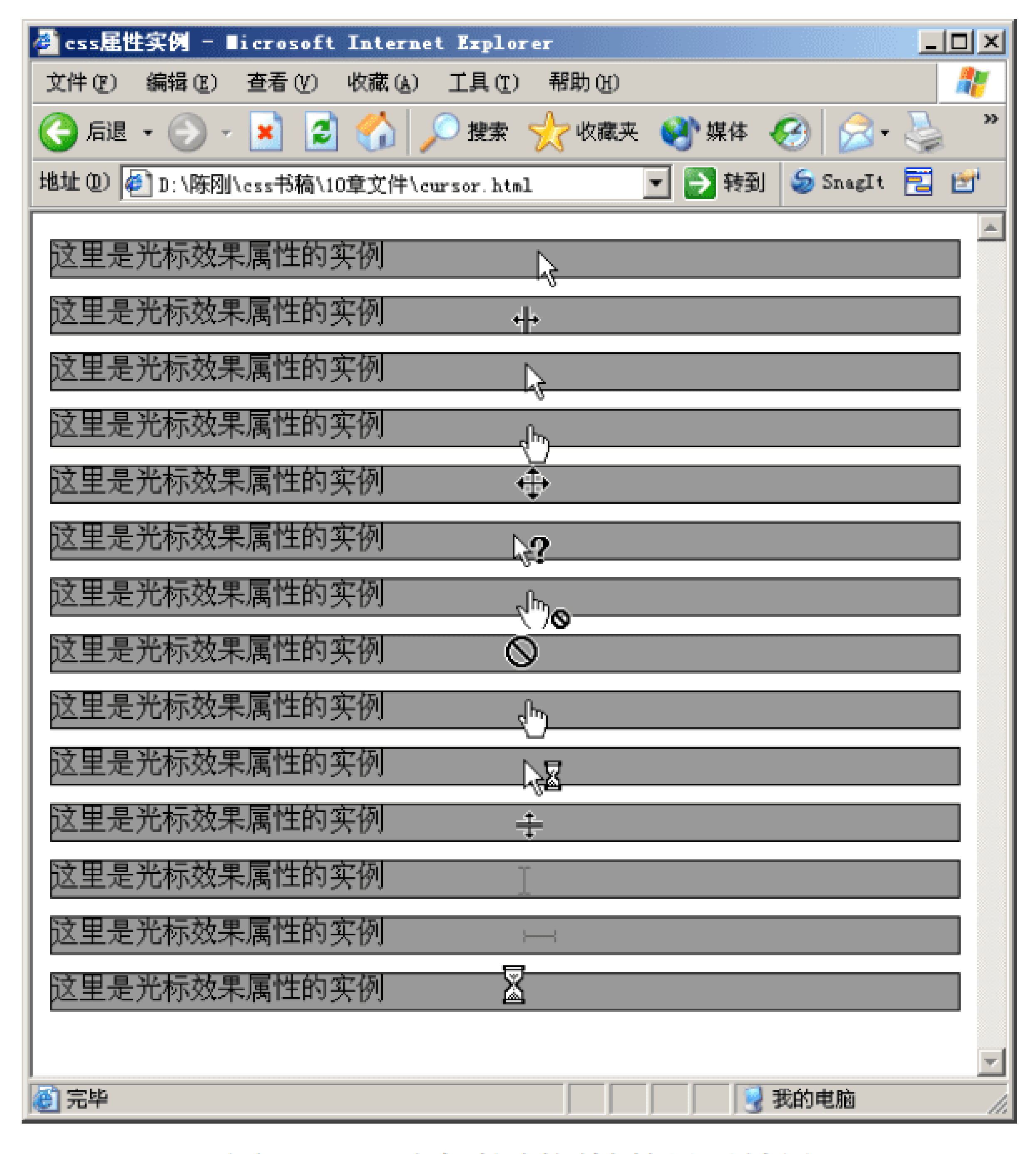


图 19-10　光标控制属性的显示效果

注意

图 19-10 是经过处理后的显示效果，在实际的页面中，无法同时显示多种光标效果。

19.3 控制元素的缩放

在 CSS 中，可以使用相应的属性，定义元素的缩放程度。当定义了缩放程度之后，元素中的所有内容，包括文本、图像等，都将按照定义的属性值进行缩放。定义元素缩放的属性是 zoom，其语法结构如下所示。

```
zoom : normal | number | percent;
```

其中部分属性值的含义如下所示。

- nomal：定义元素显示正常的大小。
- number：使用数字定义元素的缩放比例。
- precent：使用百分比数，定义元素的缩放比例。



注意

滚条的相关属性，在 IE 浏览器中才能够正常显示。

下面是一个元素缩放属性的实例，其代码如下所示。

例程 19-11 xhtml-change.html

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html;charset=gb2312" />
<title>无标题文档</title>
<style>
p{
  height:100px;
  width:300px;
  background:#cccccc;
  border:5px solid #000000;
  margin:0 0 10px 0;
  }
.p1
  {
  zoom:2;
  }
img
  {
  width:200px;
  }
</style>
</head>
<body>
```

```
<p>
<img src="pic.jpg" alt="pic" />注意缩放的效果</p>
<p class="p1">
<img src="pic.jpg" alt="pic" />注意缩放的效果</p>
12 </body>
13 </html>
```

以上的代码中，06 行在<p>元素中定义了 300*100 的方框，在<p1>元素中，定义了缩放属性值为 2。在<img>元素中定义图片的宽度为 200 像素。代码运行后的显示效果，如图 19-11 所示。

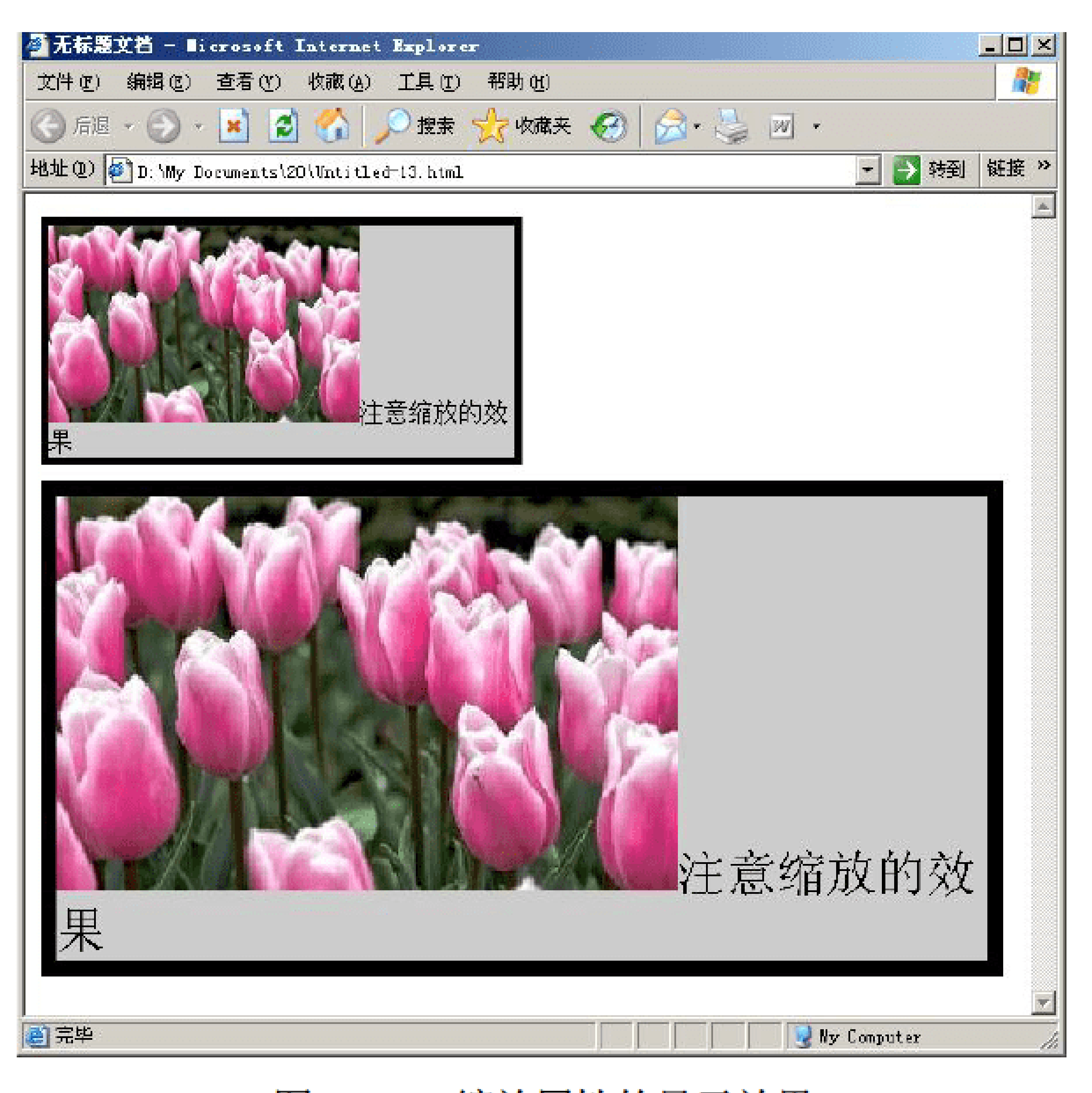


图 19-11　缩放属性的显示效果

从图 19-11 可以看出，当定义了缩放属性后，不但元素的内容会按照比例缩放，元素中定义的各种样式值也会按照比例缩放。

19.4 控制链接的显示

在网页中，通常要使用很多链接元素，定义页面的跳转。在 CSS 中，可以通过使用伪类来定义链接内容在各种状态下的显示效果。通常链接元素都有 4 个状态，未访问状态、鼠标悬停状态、链接激活状态或已访问状态，其显示效果的控制分别由 4 个伪类控制。下面进行详细讲解。

19.4.1 定义链接未访问的显示效果

定义链接未访问效果的伪类是：:link。使用:link 伪类可以定义未访问链接的各种显示效

果。:link 伪类的使用方法，如下所示。

```
:link:{属性:属性值;}
```

下面是一个使用:link 伪类的实例，其代码如下所示。

例程 19-12　:link:.html

```
01  <html xmlns="http://www.w3.org/1999/xhtml">
02  <head>
03  <meta http-equiv="Content-Type" content="text/html;charset=gb2312" />
04  <title>无标题文档</title>
05  <style>
06  a:link
  {
    color:#333333;
    font-size:24px;
    font-weight:bold;
  }
07  </style>
08  </head>
09  <body>
  <a href="#">链接的内容</a>
10  </body>
11  </html>
```

以上代码中，06 行中使用:link 伪类，定义了链接元素在未访问时的各种效果，包括文本颜色、字体大小、字体样式等。当链接内容未访问时，其显示效果如图 19-12 所示。

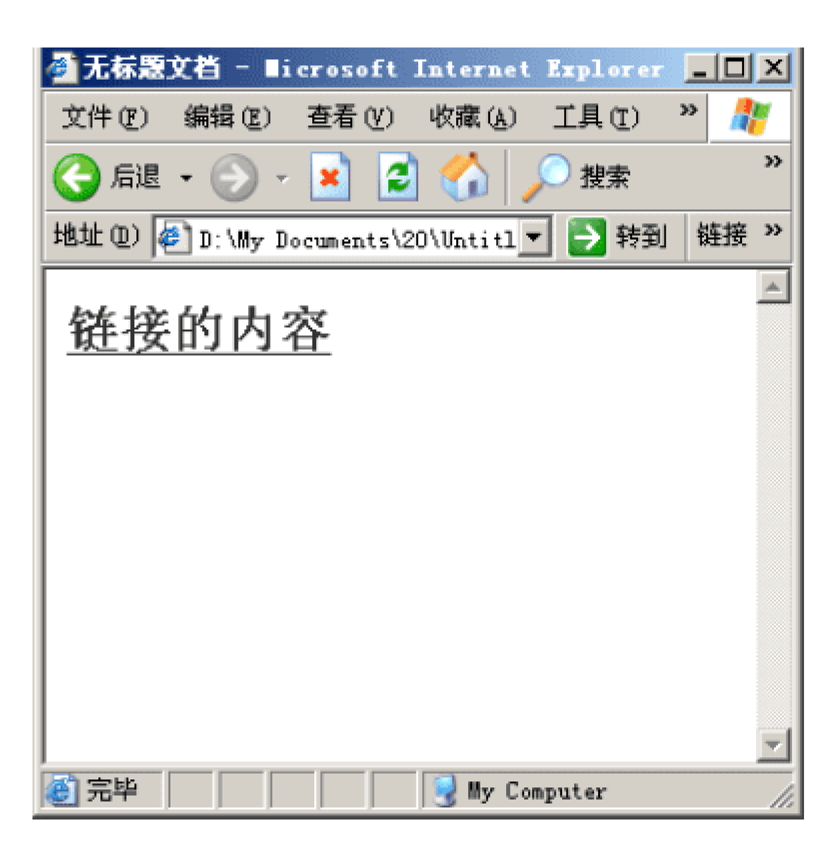


图 19-12　定义:link 伪类后链接的显示效果

19.4.2　定义链接鼠标悬停的显示效果

定义链接鼠标悬停效果的伪类是：:hover。使用:hover 伪类可以定义链接鼠标悬停的各种显示效果。:hover 伪类的使用方法，如下所示。

```
:hover:{属性:属性值;}
```

下面是一个使用:hover 伪类的实例，其代码如下所示。

例程 19-13　xhtml-change.html

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html;charset=gb2312" />
<title>css 属性实例</title>
<style>
a:hover
```

```
        {
        color:#000000;
        font-size:36px;
        font-weight:bold;
        }
07      </style>
08      </head>
09      <body>
10      <a href="#">这里是包含链接的内容</a>
11      </body>
12      </html>
```

以上代码中，06 行中使用:hover 伪类，定义了链接元素在鼠标悬停时的各种效果，包括文本颜色、字体大小、字体样式等。当鼠标悬停在链接内容上时，其显示效果如图 19-13 所示。



图 19-13　定义:hover 伪类后的显示效果

19.4.3　定义链接激活的显示效果

定义链接激活时(例如鼠标按下与释放之间)显示效果的伪类是：:active。使用:active 伪类可以定义链接激活的各种显示效果。:active 伪类的使用方法，如下所示。

```
:active:{属性:属性值;}
```

下面是一个使用:active 伪类的实例，其代码如下所示。

例程 19-14　xhtml-change.html

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html;charset=gb2312" />
<title>css 属性实例</title>
<style>
a:active
{
color:#000000;
font-size:36px;
```

```
        font-weight:bold;
        background:#999999;
        }
07      </style>
08      </head>
09      <body>
10      <a href="#">这里是包含链接的内容</a>
11      </body>
12      </html>
```

以上代码中，06 行中使用:active 伪类，定义了链接元素在激活时的各种效果，包括文本颜色、字体大小、字体样式等。当在链接元素上按下鼠标时，其显示效果如图 19-14 所示。

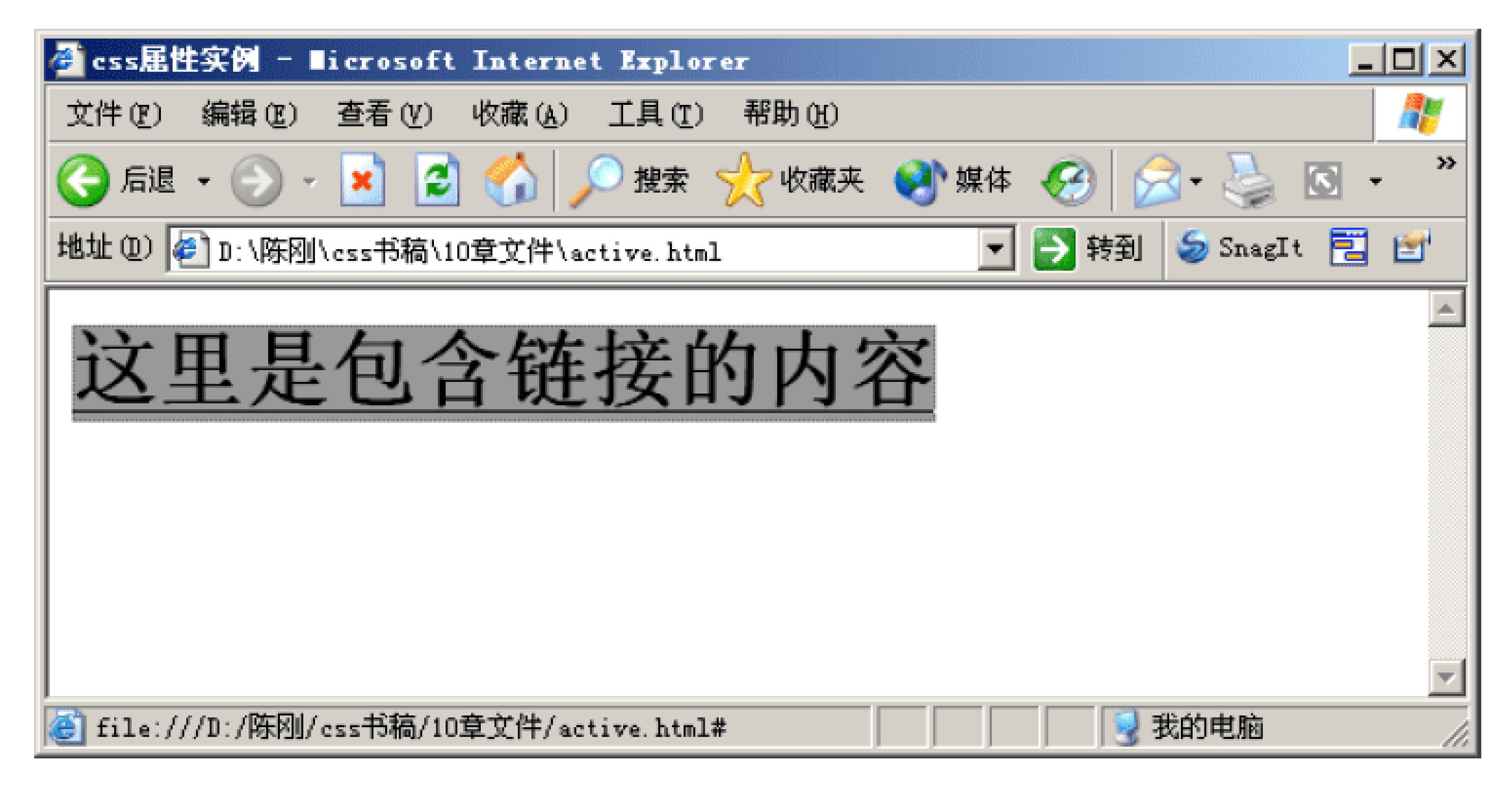


图 19-14　定义:active 伪类后的显示效果

19.4.4　定义链接访问后的显示效果

定义链接访问后显示效果的伪类是：:visited。使用:visited 伪类可以定义链接内容被访问后的各种显示效果。:visited 伪类的使用方法，如下所示。

```
:visited:{属性:属性值;}
```

下面是一个使用:visited 伪类的实例，其代码如下所示。

例程 19-15　xhtml-change.html

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html;charset=gb2312" />
<title>css 属性实例</title>
<style>
a:visited
{
color:#666666;
font-size:36px;
font-weight:bold;
text-decoration:line-through;
}
```

```
07  </style>
06  </head>
09  <body>
10  <a href="#">这里是包含链接的内容</a>
11  </body>
12  </html>
```

以上代码中，06 行中使用:visited 伪类，定义了链接元素在访问后的各种效果，包括文本颜色、字体大小、字体样式等。当链接访问后，其显示效果如图 19-15 所示。

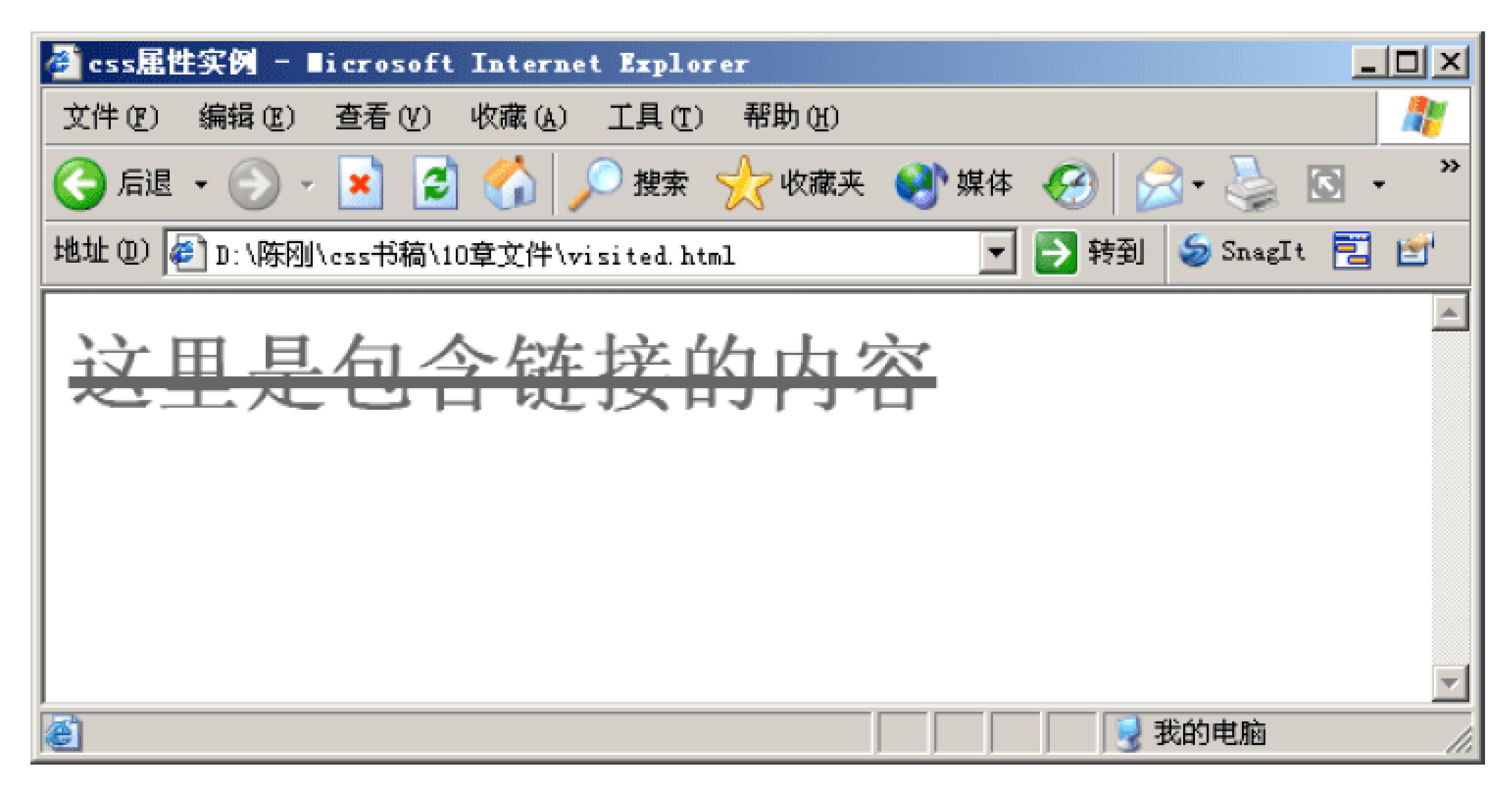


图 19-15　定义:visited 伪类后的显示效果

19.4.5　定义链接效果的顺序

在使用 CSS 属性，定义链接显示效果的时候，要按照:link、:visited、:hover、:active 的顺序定义。如果使用错误的顺序，链接属性中定义的部分效果将无法显示。

下面是一个使用 4 个伪类同时定义链接效果的实例，其代码如下所示。

例程 19-16　xhtml-change.html

```
01  <html xmlns="http://www.w3.org/1999/xhtml">
02  <head>
03  <meta http-equiv="Content-Type" content="text/html;charset=gb2312" />
04  <title>css 属性实例</title>
05  <style>
06  .a1:link
    {
    color:#666666;
    font-size:36px;
    text-decoration:none;
    }
07  .a1:visited
    {
    color:#999999;
    font-size:36px;
    text-decoration:underline;
    }
```

```
.a1:hover
    {
    color:#333333;
    font-size:48px;
    text-decoration:underline;
    }
.a1:active
  {
    color:#000000;
    font-size:36px;
    font-weight:bold;
    text-decoration:underline;
  }
.a2:link
  {
    color:#666666;
    font-size:36px;
    text-decoration:none;
  }
.a2:hover
  {
    color:#333333;
    font-size:48px;
    text-decoration:underline;
  }
.a2:active
  {
    color:#000000;
    font-size:36px;
    font-weight:bold;
    text-decoration:underline;
  }
.a2:visited
  {
    color:#999999;
    font-size:36px;
    text-decoration:underline;
  }
</style>
</head>
<body>
<a href="#" class="a1">这里是包含链接的内容</a><br />
<a href="#" class="a2">这里是包含链接的内容</a>
</body>
</html>
```

以上的代码中，制作了两个<a>元素。在每个<a>元素中，定义了相同的链接显示效果，只是使用了不同的定义顺序。

当鼠标悬停在第一个<a>元素上时，其显示效果如图 19-16 所示。

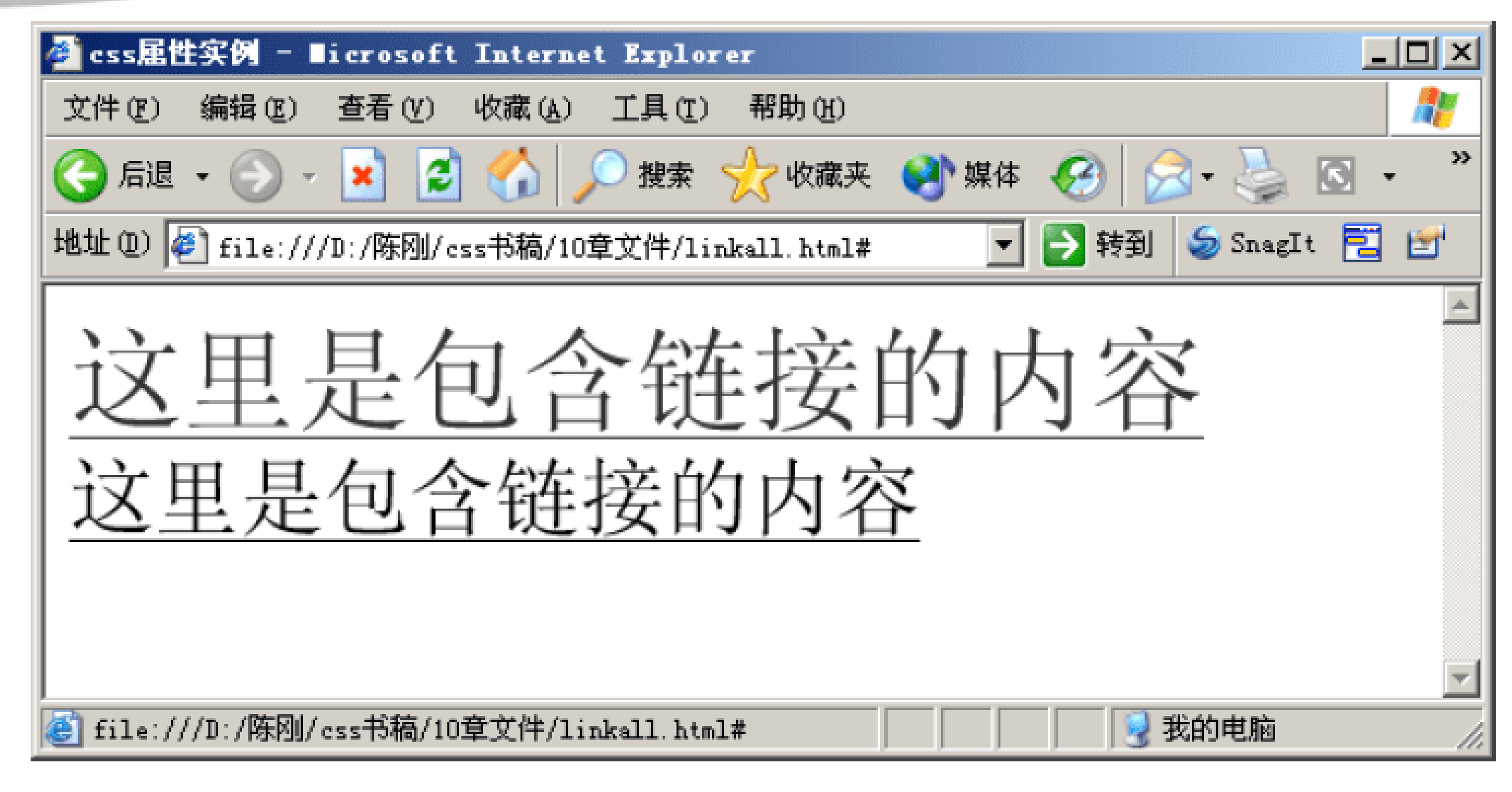


图 19-16 鼠标悬停在第一个链接元素上的效果

从图 19-16 可以看出，此时页面正确地显示了 CSS 中定义的属性。当鼠标悬停在第二个<a>元素上时，其显示效果如图 19-17 所示。

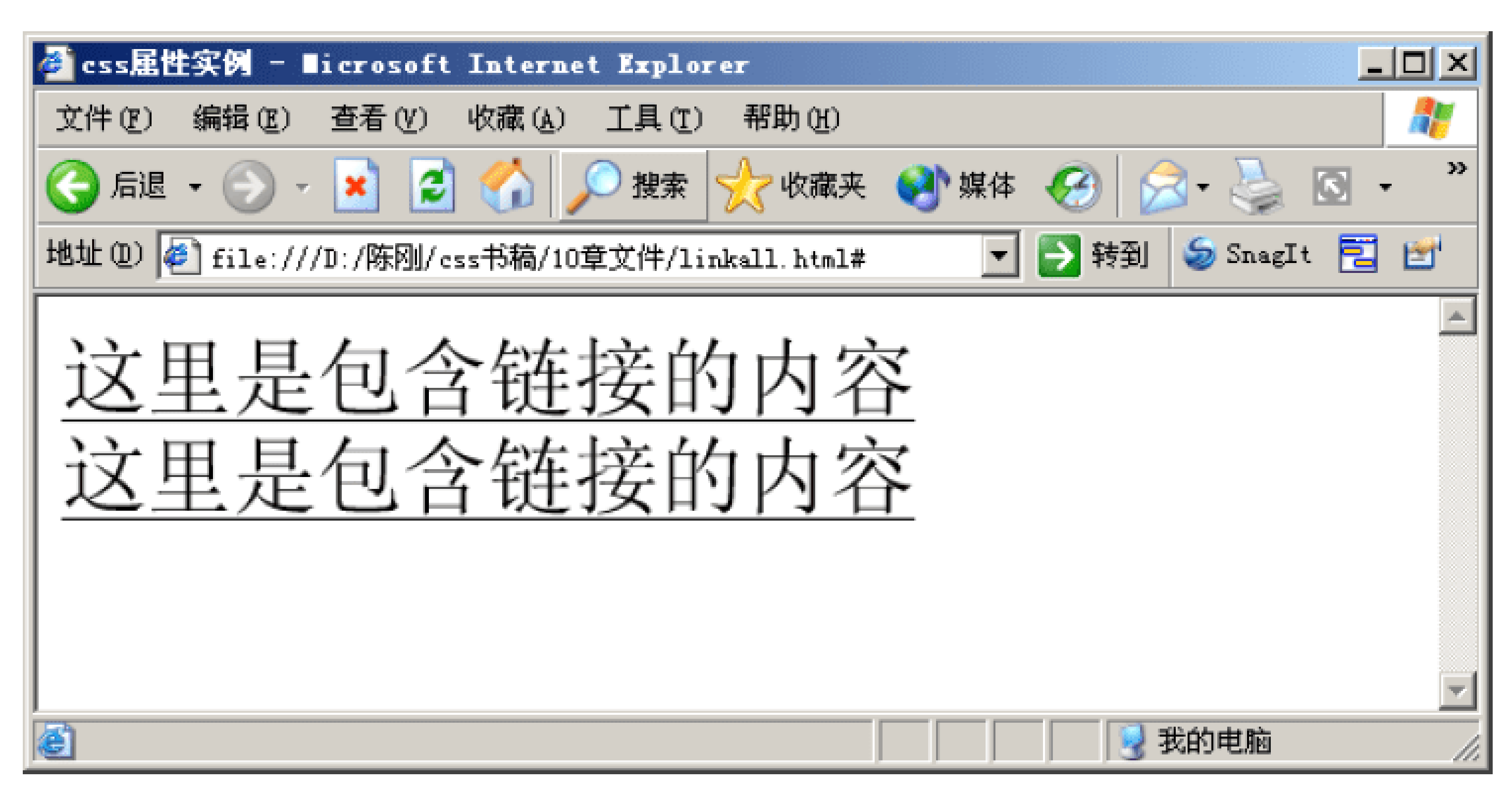


图 19-17 鼠标悬停在第二个链接元素上的效果

从图 19-17 可以看出，由于没有正确地定义链接伪类的顺序，此时页面无法正常显示鼠标悬停时的效果。

19.5 本章习题

一、选择题

1. 以下关于控制滚条的显示的说法中，错误的是：(　　)。

A. 滚条的相关属性是 IE 浏览器的特有属性。

B. 滚条的相关属性也是其他浏览器中的特有属性。

C. 滚条 3d 亮边框，是指右侧滚条中，显示在滑块和箭头区域最外层左侧和顶部的区域。

D. 单独使用滚条亮边框颜色属性时，将对滚条其他部分的颜色产生影响。

2. 以下有关滚条亮边框颜色属性的说法中，错误的是：(　　)。

A. 单独使用滚条亮边框颜色属性时，将对滚条其他部分的颜色产生影响

B. 滚条亮边框，是指右侧显示的滚条中，与滚条 3d 亮边框相邻的区域

C. 单独使用滚条亮边框颜色属性时，不会对滚条其他部分的颜色产生影响

D. 使用滚条亮边框颜色属性时，整个页面中的亮边框颜色都会变化

3．scrollbar-shadow-color 表示的意思是：(　　)。

A. 滚条箭头颜色属性

B. 滚条阴影颜色属性

C. 滚条亮边框颜色属性

D. 滚条滑块颜色属性

4．{scrollbar-face-color:yellow;}样式的作用是：(　　)。

A.滚条亮边框颜色为黄色　　B.滚条箭头颜色为黄色

C.滚条滑块颜色为黄色　　D.滚条阴影颜色为黄色

二、填空题

1．在 CSS 中光标控制属性是________，其中中间竖线，左右两个箭头的显示效果的属性是_________。手形的光标效果属性是________。

2．定义链接未访问效果的伪类是________，定义链接访问后显示效果的伪类是________。

3．在 CSS 中，定义元素的缩放程度的属性是_____，它可以使用的属性值有_______、_________和__________。

三、实战练习

1．制作页面，练习使用 CSS 样式控制滚条的显示。

2．制作页面，练习使用 CSS 样式控制光标的显示。

3．制作页面，练习使用 CSS 样式控制链接的显示。

Chapter 20

制作个人博客页面

个人博客页面用来展示个人的一些信息、日志以及个人的图片等，是一个方便的展示平台。个人博客页面的设计和制作都比较自由。具体的制作过程包括以下几个步骤：制作页面效果图，这个部分通常使用 Photoshop 完成；然后再使用 Photoshop 的切片工具，将效果图的各个部分制成图片，最后在 Dreamweaver 中制作成网页。

本章主要内容有：

- ◎ 掌握网站页面的制作流程。
- ◎ 学会合理地划分和规划页面。
- ◎ 学会合理地公用代码。
- ◎ 学会规划站点的样式文件。
- ◎ 学会优化代码和页面结构。

20.1 制作页面前的准备工作

在制作具体页面之前，先要对页面进行大体的规划，这个规划可以帮助确定页面的大体结构，这个步骤一般在头脑中进行，可以不必显示在效果图上。在页面区块大概构思完成后，使用 Photoshop 的切片工具对页面进行切割，在切割的过程中制作出页面的背景等修饰图片。

20.1.1 规划页面的内容

页面结构的规划是制作整个页面的基础，好的页面规划能够使页面更具有扩展性，能够适应页面内容的变化。在规划页面时，首先分割页面为几个部分，如 Logo、Banner、导航条、侧栏等内容。然后在各个部分中切出内容和背景。

本实例中效果图的大概划分结构如图 20-1 所示。

在如图 20-1 所示的结构图中，将整个页面纵向分成了 4 个部分，分别用来显示主要的几个内容。这样划分的目的是让页面具有更大的伸缩性。



图 20-1 页面结构图

20.1.2 切分效果图

在 Photoshop 中首先区分页面中的修饰图片和内容图片。然后将页面中的文本内容等隐藏，根据构思的页面区块，切割出背景图片和内容图片。最后将各种图片内容保存在磁盘的某个文件夹中。

从软件保存后的图片，都会默认地使用编号的格式定义图片的名称。在实际使用时，这些名称不具有实际的含义，建议将图片名称修改为与内容相关。最终保存的图片内容如图 20-2 所示。



图 20-2 切割保存后的图片文件

20.2 规划站点文件夹

准备好各种页面文件之后，需要做的是制作好站点的各种文件夹。通常各种图片文件都放在名称为 images 的文件夹中。样式表文件单独放在一个名称为 style 的文件夹中。站点的首页一般命名为“default”，并保存在根文件夹下。此时，站点的目录结构如图 20-3 所示。

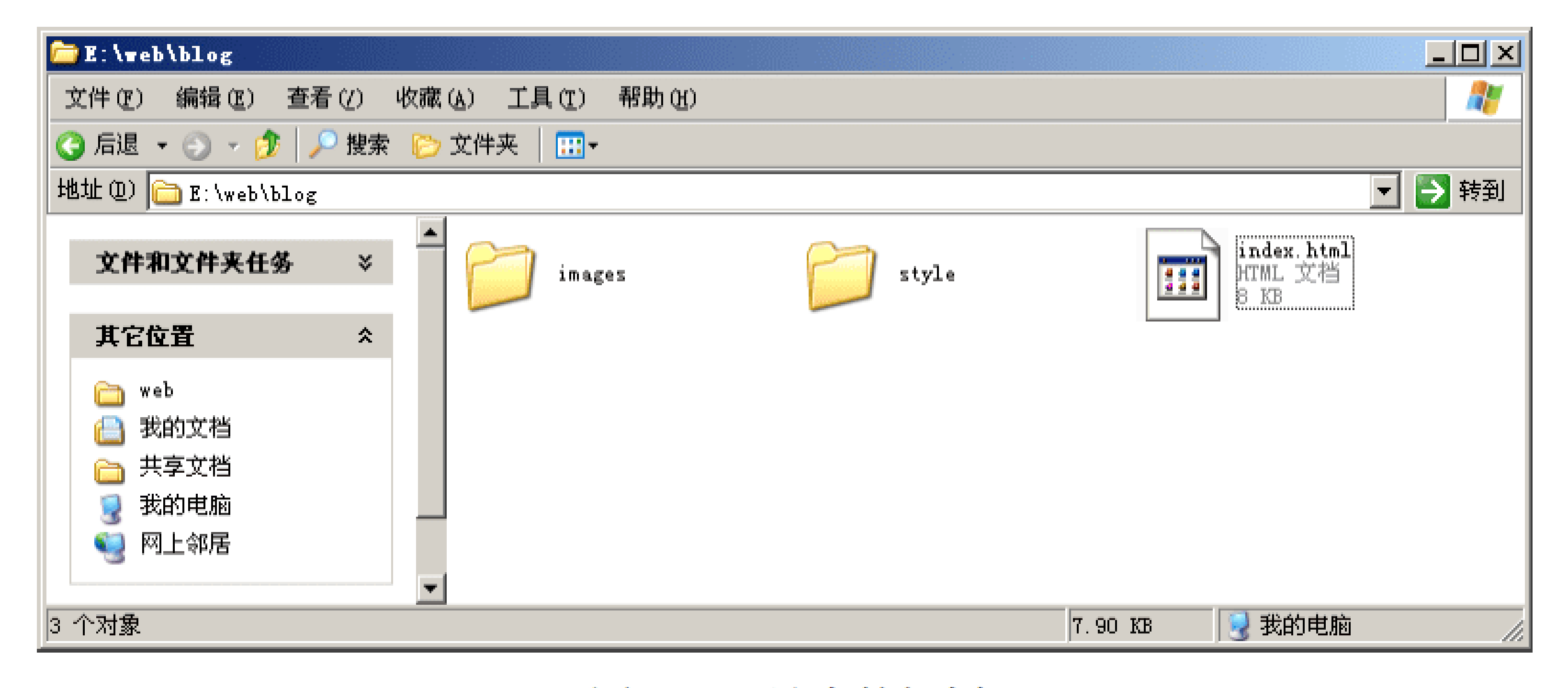


图 20-3 站点的规划

20.3 定义基本的样式

在站点建立后，就可以制作 CSS 样式文件，并关联在 XHTML 文件中。然后通过在 XHTML 中，对每个元素定义(或关联)不同的样式，制作页面的各个部分。按照最初规划的页面结构，整个页面分为头部内容、导航内容、主体内容、底部内容几个部分。本节讲解页面基础样式的制作，其具体内容如下所示。

20.3.1 新建 CSS 文件

在制作具体的内容之前，首先要制作好需要使用的 CSS 样式文件，并在 XHTML 中调用该文件。这样做可以将页面结构部分和表现修饰部分分离到两个文件中，便于后期的维护。在 style 文件夹中右击新建一个文本文档，然后更改名称为“style.css”。

20.3.2 定义页面的基础样式

新建页面之后，页面的默认标题为“无标题文档”，所以还需要将文档更改为更有意义的名称。另外为了更好地显示页面内容，以及辅助宣传页面，还需要修改和定义页面的文字编码、<meta>标签等相关内容，其具体的代码如下所示。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gbk" />
<title>一梦如是-博客</title>
<meta name="Keywords" content="博客,Blog,Blogger,中飞,Sohu,网络日志  手心的太阳" />
<meta name="description" content="一梦如是  无端天与娉婷，夜月一帘幽梦" />
<meta name="robots" content="index,follow" />
<link rel="stylesheet" type="text/css" href="style/style.css" />
</head>
```

以上代码中，使用<meta>元素定义了页面的各种信息，然后使用<link>元素定义了页面关联的样式文件“style.css”。在关联的 CSS 样式文件中，定义了页面使用的基础样式，其代码如下所示。

```
body {
    margin: 0px;                                                    /*定义边界的综合属性为 0*/
    padding: 0px;                                                   /*定义主体的补白属性为 0 */
    font-family: "宋体",Verdana, Arial, Helvetica, sans-serif;      /*定义主体部分的字体为宋体*/
    font-size: 12px;                                                /*定义主题部分的字体大小为 12 像素*/
    color: #cccccc;                                                 /*定义主体字体的颜色为浅灰色*/
    line-height: 130%                                               /*定义行高属性为正常的 1.3 倍*/
```

```
        background-color: #11191E;                    /*定义主体背景的颜色的黑色 */
    }
    a {
        color: #F5651F;                               /*定义字体的颜色为橘红色*/
        text-decoration: underline;                   /*定义文字下划线*/
    }
    a:hover {
        color: #f4f4f4;                               /*定义字体的颜色为白色*/
        text-decoration: underline;                   /*定义文字下划线*/
    }
    img {
        border: none;                                 /*图片无边框*/
    }
```

在页面的基础样式中，通过 body 选择符定义了页面使用的字体大小、文字的颜色、行高、背景颜色等。这些样式是页面整体效果以及通用的设置。在 a 选择符中定义了页面链接的颜色，配合:hover 伪类可以定义这个页面链接的显示效果。最后定义图片的边框为 0，目的是消除图片默认边框带来的显示差异。

20.4 制作页面头部

定义了页面基础样式后，就可以开始进行页面头部内容的制作，其具体内容如下所示。

20.4.1 制作页面头部的结构

从效果图可以看到，页面头部的内容为两行文本，其中部分文本包含超级链接，为了更好地独立控制每个部分的显示效果，在制作结构时为各种内容定义了不同的 id，便于分别控制每个部分的显示效果，其具体的代码如下所示。

```
<div id="wrapper">
    <div id="innerWrapper">
      <div id="header">
        <div id="innerHeader">
          <h1 id="blogTitle"><a href="#">明天会更好</a></h1>
          <div id="blogUrl"><a href="#">http://abc.*****.com/</a> <a href="#">复制地址</a> </div>
          <div id="blogDesc">无端天与娉婷，夜月一帘幽梦</div>
        </div>
    </div>
</div>
```

20.4.2 定义页面头部的样式

根据页面头部的结构，可以分析出头部的代码分为几个部分，一部分是整体定义页面大小

和背景的 wrapper 样式。其具体的代码如下所示。

```
#wrapper {
        margin: 0px auto;                                          /*边界综合属性设置为自动*/
        width: 993px;
        background-image: url(../images/sohu10_01.jpg);            /*调用背景图片*/
        background-position: left top;                             /*背景图片位置置于左上*/
        background-repeat: no-repeat;                              /*背景图片不重复*/
}
```

以上代码中，主要定义了页面头部使用的背景图片，并定义了背景图片的显示方式和位置。接下来是页面头部内容的部分，首先是关于头部内容位置的确定，其代码如下所示。

```
#header {
        height: 374px;                         /*定义头部的高度的为 374 像素*/
}
#innerHeader {
        width: 983px;
        padding-left: 10px;                    /*定义左侧补白 10 像素*/

}
```

在上面的代码中主要定义了头部元素占有空间的大小，以及与左侧边界之间的距离。当这些确定之后，就可以进一步定义内容的显示方式和位置了，其具体的代码如下所示。

```
#blogTitle {
        float: left;                           /*定义字体向左浮动*/
}
#blogTitle, #blogTitle a {
        color: #ffffff;                        /*定义字体颜色为白色*/
        font-size: 16px;                       /*定义字体大小为 16 像素*/
        line-height: 16px;                     /*定义行高为 16 像素*/
text-decoration: none;                         /*定义字体无修饰*/
        margin-top: 15px;                      /*定义顶部边界宽度为 15 像素*/
        padding-left: 70px;                    /*定义左侧补白为 70 像素*/
}
#blogTitle a:hover {
        color: #F5651F;                        /*定义字体颜色为橘红色*/
}
#blogUrl {
        color: #ffffff;                        /*定义字体颜色为白色*/
        padding-top: 15px;                     /*定义顶部补白为 15 像素*/
}
#blogUrl a {
        color: #ffffff;                        /*定义字体颜色为白色*/
        padding-left: 10px;                    /*定义四边补白为 70 像素*/
}
#blogUrl a:hover {
        color: #F5651F;                        /*定义字体颜色为橘红色*/
}
```

```
#blogDesc {
    color: #ffffff;                    /*定义字体颜色为白色*/
    width: 400px;                      /*定义文本的宽度为 400 像素*/
    word-break: break-all;             /*定义文本中可以在单词中换行显示*/
    padding-top: 15px;                 /*定义顶部补白 150 像素*/
    padding-left: 140px;               /*定义左侧补白 140 像素*/
}
```

在上面定义的 CSS 样式中，主要定义了头部标题显示的位置，以及超链接的内容在鼠标悬停与未单击之前的显示效果。其中元素位置的确定是通过使用浮动属性、边界属性和补白属性一起完成的。代码运行后页面的显示效果如图 20-4 所示。

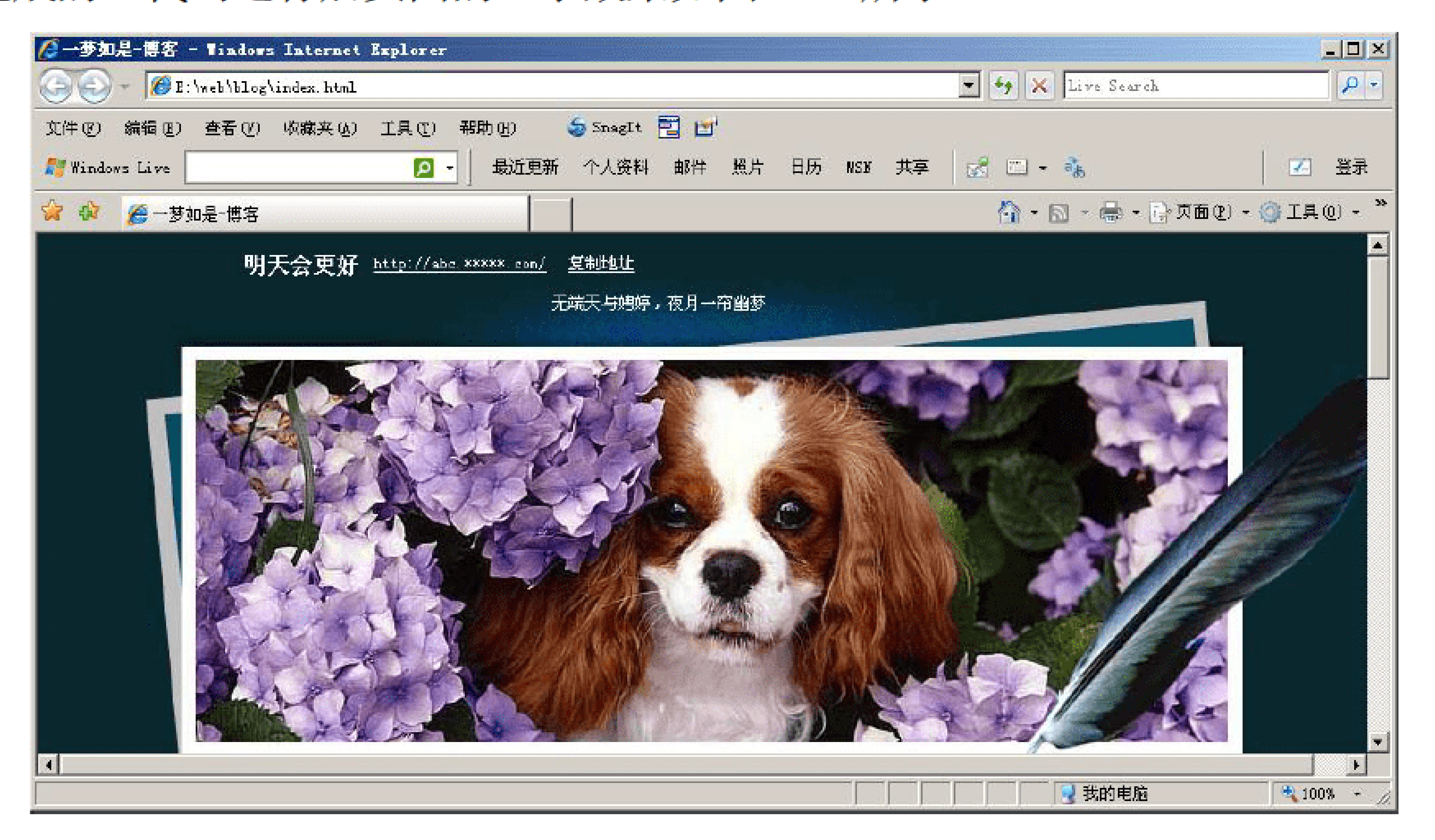


图 20-4　页面头部的显示效果

20.5 制作页面导航

页面导航内容包括制作导航的结构、导航的背景、链接样式等。在导航的样式中，首先要对各个元素进行精确定位。然后使用各种伪类和类选择符，制作出每个导航链接的独立背景，以及显示、隐藏效果。其具体内容如下所示。

20.5.1 制作页面导航的结构

页面导航条主要由一个<div>元素和一个<ul>元素嵌套而成。其中<div>元素用来制作导航部分的背景，<ul>元素用来制作导航条的具体内容。其具体的 XHTML 代码如下所示。

```
<div id="navBar">
    <div id="innerNavBar">
        <ul id="mainNav">
        <li><a href="#" class="navHome">首页</a></li>
```

```
            <li><a href="#" class="navBlog">日志</a></li>
            <li><a href="#" class="navSound">音乐</a></li>
          <li><a href="#" class="navImg">相册</a></li>
            <li><a href="#" class="navProfile">档案</a></li>
            <li><a href="#" class="navfriend">交友</a></li>
            <li><a href="#" class="navshipin">视频</a></li>
            </ul>
            <div class="clear"></div>
        </div>
</div>
```

在上面的 XHTML 结构中，每个<li>元素代表一个独立的导航内容。通过在每个<li>元素中定义不同的类属性，可以控制每个导航内容的显示效果。

20.5.2 定义页面导航的样式

导航的各种表现效果，主要通过在导航条的各个元素中定义背景和伪类实现的。大体可以分为两个部分，一部分用来定义导航内容的位置和整体效果，另一部分用来定义每个导航鼠标悬停时候的转换效果。其中导航内容位置以及整体控制的样式如下所示。

```
#navBar {
    height: 82px;                                         /*定义高度为 82 像素*/
    background-image: url(../images/sohu10_02.gif)       /*添加背景图片*/;
    background-repeat: no-repeat;                         /*背景图片不重复*/
}
#innerNavBar {
    width: 710px;                                         /*定义文本宽度为 710 像素*/
    margin-left: 170px;                                   /*定义左侧边界属性值为 170px*/
    margin-top: 23px;                                     /*定义顶部边界属性值为 23px*/
}
#mainNav {
    float: left;                                          /*定义文本内容左飘移*/
    margin: 0px;                                          /*定义边界综合属性为 0*/
    padding: 0px;                                         /*定义综合补白属性为 0*/
}
#mainNav li {
    float: left;                                          /*定义文本内容左飘移*/
    list-style: none;                                     /*定义不使用列表符号*/
    margin: 0px;                                          /*定义边界属性为 0 像素*/
    padding: 32px 0px 2px;                                /*定义综合补白为上方 32 像素，下方 2 像素*/
}
#mainNav li a {
    padding: 32px 12px 0px;                               /*定义综合补白为上方 32 像素，左侧 12 像素*/
    background-position: center top;                      /*定义背景图片居中靠上显示*/
    background-repeat: no-repeat;                         /*定义背景图片不重复*/
    text-decoration: none;                                /*定义字体无修饰*/
}
```

```
#mainNav li a {
    color: #ffffff;                                    /*定义文本字体为白色*/
}
```

在上面的代码中分别定义了导航条的高度、宽度，每个导航内容之间的距离以及导航条中文本所使用的颜色等各种内容。为了让导航条显示的内容更加丰富，同时在鼠标悬停时可以做出反应，还需要对每个链接进行进一步的定义，具体代码如下所示。

```
#mainNav .navHome { background-image: url(../images/icoMenu_home.gif);}
#mainNav .navHome:hover { background-image: url(../images/icoMenu_home_over.gif);}
#mainNav .navHome-active { background-image: url(../images/icoMenu_home_over.gif); color: #ffffff; cursor: default;}
#mainNav .navBlog { background-image: url(../images/icoMenu_blog.gif);}
#mainNav .navBlog:hover { background-image: url(../images/icoMenu_blog_over.gif);}
#mainNav .navBlog-active { background-image: url(../images/icoMenu_blog_over.gif); color: #ffffff; cursor: default;}
#mainNav .navImg { background-image: url(../images/icoMenu_img.gif);}
#mainNav .navImg:hover { background-image: url(../images/icoMenu_img_over.gif);}
#mainNav .navImg-active { background-image: url(../images/icoMenu_img_over.gif); color: #ffffff; cursor: default;}
#mainNav .navSound { background-image: url(../images/icoMenu_sound.gif);}
#mainNav .navSound:hover { background-image: url(../images/icoMenu_sound_over.gif);}
#mainNav .navSound-active { background-image: url(../images/icoMenu_sound_over.gif); color: #ffffff; cursor: default;}

#mainNav .navVideo { background-image: url(../images/icoMenu_video.gif);}
#mainNav .navVideo:hover { background-image: url(../images/icoMenu_video_over.gif);}
#mainNav .navVideo-active { background-image: url(../images/icoMenu_video_over.gif); color: #ffffff; cursor: default;}

#mainNav .navProfile { background-image: url(../images/icoMenu_profile.gif);}
#mainNav .navProfile:hover { background-image: url(../images/icoMenu_profile_over.gif);}
#mainNav .navProfile-active { background-image: url(../images/icoMenu_profile_over.gif); color: #ffffff; cursor: default;}

#mainNav .navshipin { background-image: url(../images/video.gif);}
#mainNav .navshipin:hover { background-image: url(../images/videoblack.gif);}
#mainNav .navshipin-active { background-image: url(../images/videoblack.gif); color: #ffffff; cursor: default;}

#mainNav .navfriend { background-image: url(../images/friend.gif);}
#mainNav .navfriend:hover { background-image: url(../images/friendblack.gif);}
#mainNav .navfriend-active { background-image: url(../images/friendblack.gif); color: #ffffff; cursor: default;}
```

在上面的代码中，每个导航内容使用的样式原理是相同的。首先为元素定义一个背景图片，然后定义当鼠标悬停时，背景图片转换为另一个，这样就实现了一种动态转换的效果。定义导航内容的样式后，页面的显示效果如图 20-5 所示。



图 20-5　导航条的显示效果

20.6 制作页面主体

页面主体内容由日志内容和侧栏内容两个部分组成。由于日志和侧栏部分的内容都有可能扩展，所以在制作时要将高度定义为自动伸展。由于日志部分和侧栏部分的内容都很多，所以本节省略了这两部分的制作，只讲解主体结构的制作方法。

20.6.1　制作页面主体的结构

在页面的主体内容中，由于导航部分的背景高度比较高，所以要使用负的边界值，将内容向上移动。由于内容位置的原因，在日志部分和侧栏部分需要使用更复杂的嵌套结构，其具体的 XHTML 代码如下所示。

```
<div id=" mainWrapper ">
   <div id="content">
      <div class="innercontent">
  <!--==========日志部分============-->
      </div></div>
   <div id="sidebar">
      <div id="innersidebar">
<!--==========侧栏部分============-->
      </div></div>
   <div class="clear"></div>
</div>
```

在上面的 XHTML 结构中，innercontent 和 innersidebar 这两个元素，用来移动亮部分的位置。由于 content 元素和 sidebar 元素需要使用浮动属性同行显示，所以需要添加一个辅助的 clear 元素，去除浮动带来的影响。

20.6.2　定义页面主体内容的样式

页面主体内容主要由页面主体使用的背景图片，以及两个主要内容的位置显示效果构成的。其具体的代码如下所示。

```
#mainWrapper {
    background-image: url(../images/sohu10_04.gif);        /*定义添加背景图片*/
    background-repeat: repeat-y;                           /*定义背景图片不重复*/
    width: 993px;                                          /*定义文本宽度为 993 像素*/
}
#sidebar {
    float: right;                                          /*定义文本右飘移*/
    width: 181px;                                          /*定义文本宽度为 181 像素*/
    overflow: hidden;                                      /*定义了溢出内容显示效果为隐藏*/
    margin-top:-10px;                                      /*定义了顶部边界属性为－10 像素*/
    padding-right: 120px;                                  /*定义了右侧补白属性为 120 像素*/
}
#content {
    float: left;                                           /*定义文本左飘移*/
    width: 530px;                                          /*定义文本宽度为 530 像素*/
    overflow: hidden;                                      /*定义了溢出内容显示效果为隐藏*/
    margin-top:-20px;                                      /*定义了顶部边界属性为－20 像素*/
    padding-right: 125px;                                  /*定义了右侧补白属性为 125 像素*/
}
```

在上面的代码中，使用 mainWrapper 选择符定义了页面主体部分使用的背景图片，以及重复的方式，同时定义了主体部分的宽度。在侧栏和内容的部分使用浮动属性定义了各自的显示位置，同时通过定义相应的宽度，使两个区域处于合理的位置。接下来定义两个区域内容的显示大小和方式。

```
#innerContent {
    width: 500px;                                  /*定义文本宽度为 500*/
    overflow: hidden;                              /*定义了溢出内容显示效果为隐藏*/
}
#innerSidebar {
    overflow: hidden;     /*定义了溢出内容显示效果为隐藏*/
}
```

在上面的代码中，主要定义了内容超出元素范围时会被隐藏，这样做可以保证页面显示效果的正常。定义页面主体内容样式后，就可以在其中制作需要的各种内容了。

20.7 制作日志

日志内容主要由几个重复的结构完成，其中需要注意的是：控制各个区域的分隔距离。另外由于日志部分是由几个部分组成的，所以还要为每个部分定义文本不同的显示颜色。日志内容的具体制作过程如下所示。

20.7.1　制作日志内容的结构

日志结构分为 3 个部分：日志标题、日志内容和底部链接。由于要对 3 个部分进行分隔和修饰，所以要各自使用独立的元素定义，其具体的 XHTML 代码如下所示。

```
<div id=diarytitle><img src="images/spacer.gif" alt="" class="arrow-up"   /> 2009-10-20 </div>
  <div id=diarycontent>很多很黑的夜里，很多很亮的街灯，弯曲着通向远方；很多很凉的清晨，很多很忙的脚步，匆匆的走向理想。许多的挣扎，许多的努力，不过为了引起别人的注意；许多的错误，许多的愚蠢，只是为了坚守完整的自己。世界从来不会因此而改变，明天也不会因此变的更好。我们从来不曾得到，也没有什么可以失去。生命不过永恒黑暗中，如惊鸿般灿烂的一瞬。静静的体会你生命中的每一分钟，所有的欢笑，所有的痛苦，所有的痴心无望的追求，所有的天真纯净的笑容，构成了你的一生。 </div>
  <div id=diarybottom><a href="#">评论 (0)</a>  |<a href="#">固定链接</a> |  <a href="#">类别 (闲时随笔) </a>|  发表于 15:44 </div>
```

在上面的 XHTML 结构中，使用 3 个<div>元素分别定义日志的 3 个部分内容，每个元素之间相互独立。

20.7.2　定义日志内容的样式

日志部分由 3 个部分组成，日志的标题、日志的内容、日志的相关信息。在每个部分中都需要定义文本的显示方式和位置，其具体的代码如下所示。

```
#diarytitle{
    width: 100%;                          /*定义文本内容全屏显示*/
    float:left;                           /*定义文本左漂移*/
    font-size:14px;                       /*定义文本字体大小为 14 像素*/
    font-weight:bold;                     /*定义文本字体为黑体*/
    color:#eeeeee;                        /*定义字体为白色*/
    padding: 5px;                         /*定义文本综合补白为 5 像素*/
    border-bottom: 1px dashed #97ACB6;    /*定义底部边框为线性宽度为 1 像素*/
}
#diarycontent{
    width: 100%;                          /*定义文本内容全屏显示*/
    float:left;                           /*定义文本左漂移*/
    font-size:12px;                       /*定义文本字体大小为 12 像素*/
    color:#dddddd;                        /*定义字体颜色为浅灰色*/
    padding: 5px;                         /*定义综合补白距离是 5px*/
    line-height:180%;                     /*定义行高为字体的 1.8 倍*/
    padding-bottom:15px;                  /*定义底部补白距离是 15px*/
}
#diarybottom{
    width: 100%;
    float:left;                           /*定义文本左漂移*/
    font-size:12px;                       /*定义文本字体大小为 12 像素*/

    color:#eeeeee;                        /*定义字体为白色*/
    padding: 5px;                         /*定义上下左右的补白距离是 5px*/
}
```

在上面的代码中分别定义了 3 个部分文本的大小、颜色以及行高等各种属性，然后通过定义边线的方法制作了日志内容之间的分割线。定义日志内容样式后，页面的显示效果如图 20-6 所示。



图 20-6　日志内容的显示效果

20.8 制作侧栏

侧栏内容用来显示相关的个人信息、友情链接、日志分类等内容。在这部分内容中，CSS 样式包括两个部分，一部分是通用的统一样式，另一部分是各个内容的独立样式。侧栏内容的具体制作方法如下所示。

20.8.1 制作侧栏的结构

侧栏内容包括：个人档案、归档信息、友情链接、日志分类、推荐日志、统计信息等几个部分。其中每个部分都是由标题、内容两个部分构成，根据内容的不同，所使用的结构略有区别。侧栏部分定义的 XHTML 代码如下所示。

```
    <!--个人档案 -->
          <div class="panel">
            <div class="panel-title">
            <h3><img src="images/spacer.gif" alt="" class="arrow-up" /> 个人档案</h3>
              <div class="clear"></div></div>
        <div class="panel-menu"></div>
            <div class="panel-content">
            <div class="show">
            <img src="images/show.jpg" alt="用户头像" /></div>
        <hr />
         <div class="profile">
                    昵称：倩倩<br />位置：北京市<br />个性介绍： <br />开心快乐的过每一天，锻炼身体，
```

```
努力减肥！爸爸，妈妈&hellip;我好想家！<br />
        </div></div></div>
    <!--归档 -->
    <div class="panel">
    <div class="panel-title">
    <h3><img src="images/spacer.gif" alt="" class="arrow-up"   /> 归档</h3>
    <div class="clear"></div>
    </div>
    <div class="panel-content">
    <ul>
            <li><a href="#">2009 年 8 月</a></li>
            <li><a href="#">2009 年 7 月</a></li>
            <li><a href="#">2009 年 6 月</a></li></ul>
    </div></div>
    <!--友情链接 -->
    <div class="panel">
      <div class="panel-title">
        <h3><img src="images/spacer.gif" alt="" class="arrow-up" /> 友情链接</h3>
        <div class="clear"></div></div>
        <div class="panel-content">
          <div>暂无链接</div></div></div>
    <!--日志分类 -->
    <div class="panel">
        <div class="panel-title">
         <h3><img src="images/spacer.gif" alt="" class="arrow-up"   /> 日志分类</h3>
          <div class="clear"></div></div>
        <div class="panel-content">
    <ul>
          <li><a href="#">我的心情</a></li>
          <li><a href="#">闲时随笔</a></li>
          <li><a href="#">工作感悟</a></li></ul>
        </div></div>
    <!--更新的博客 -->
    <div class="panel">
        <div class="panel-title">
         <h3><img src="images/spacer.gif" alt="" class="arrow-up"   /> 更新的博客</h3>
          <div class="clear"></div></div>
        <div class="panel-content" style="width: 175px;">
    <div class="collect"><a href="#" target="_blank" title="美丽新世界"><img src="images/diary1.jpg" alt="美丽
新世界" /> <span>美丽新世界</span></a></div><div class="collect"><a href="#" target="_blank" title="记住永远
"><img src="images/diary2.jpg" alt="告别忧伤" /> <span>记住永远</span></a></div><div class="collect"><a
href="#" target="_blank" title="爱的小屋"><img src="images/diary3.jpg" alt="爱的小屋" /> <span>爱的小屋
</span></a></div></div></div>
    <!--统计信息 -->
    <div class="panel">
        <div class="panel-title">
         <h3><img src="images/spacer.gif" alt="" class="arrow-up"   /> 统计信息</h3>
         <div class="clear"></div></div>
        <div class="panel-content">
         <ul>
```

```
            <li>创建：2009-09-27</li>
            <li>文章：11 篇</li>
            <li>评论：5 篇</li>
            <li>访问：<br />
    </ul></div></div>
```

在上面的 XHTML 结构中，使用注释的方式，标记了代码的各个部分。在每个部分中，标题和文本都定义了相同的类，具体的内容部分，由于各自独立的特点，所以定义了独立的样式。

20.8.2 定义侧栏的通用样式

侧栏的通用样式包括侧栏标题样式、侧栏文本样式、侧栏链接样式等。这些样式都是每个侧栏内容都要使用的样式。其中侧栏每个内容的整体控制，以及侧栏标题的显示效果是通过下面的代码完成的，具体内容如下所示。

```
.panel {
    margin-bottom: 5px;             /*定义下外边距为 5 像素*/
    line-height:20px;               /*定义行高为 20 像素   */
}
.panel-title{
    color:#F4F4F4;                  /*定义文本颜色*/
    height: 20px;                   /*定义高度为 20 像素*/
    padding-top: 8px;               /*定义顶部边距为 8 像素*/
}
.panel-title h3 {
    font-size: 14px;                /*定义文本字体大小为 14 像素*/
    float: left;                    /*定义文本左浮动*/
    margin: 0px;                    /*定义所有外边距为 0*/
    padding-top: 3px 0px 0px 5px;   /*定义上外边距为 3 像素、左和下外边距为 0 像素*右外边距为 5*/
}
.panel-title h3 a {
    font-size: 14px;                /*定义字体大小为 14 像素  */
}
.panel-menu{
    padding-right: 3px;             /*定义了右侧补白属性为 3 像素*/
    padding-left: 3px;              /*定义了左侧补白属性为 3 像素*/
    float: right;                   /*定义了文本由浮动*/
}
```

接下来定义侧栏内容的显示效果，内容主要由一些列表和相应的链接组成的，其中定义的具体代码如下所示。

```
.panel-title{
    color:#F4F4F4;                  /*定义文本颜色为白色*/
    height: 20px;                   /*定义高度为 20 像素*/
    padding-top: 8px;               /*定义顶部边距为 8 像素*/
}
```

在上面的代码中，分别定义了内容显示的空间、颜色、鼠标悬停效果、分割线、图片的显示方式、列表的显示效果等。定义完相应的样式后，页面的显示效果如图 20-7 所示。



图 20-7　定义侧栏样式后的显示效果

20.8 制作页面底部内容

页面底部内容包括站点的相关信息、邮箱地址、欢迎口号等。在制作页面底部内容的时候，只需要对各个文本和图片元素进行定位，同时定义好链接文本的显示效果即可。制作页面底部内容的具体步骤如下所示。

20.9.1　制作页面底部的结构

页面底部内容主要由一些文本内容构成，在这里为了方便，使用表格制作了各种内容。其具体的代码如下所示。

```
<div id="footer">
    <div id="innerFooter"><table id="contact"><tr><td rowspan="2"></td><td nowrap="nowrap" align="left"><a href="#" target="_blank">客服论坛</a>    客服邮箱：<a href="mailto:#">*******@*****.com</a></td></tr><tr><td nowrap="nowrap" align="left">24 小时客服热线:****-*******(人工 8:00-24:00)</td></tr></table><div id="copyright">Copyright &copy; 2009 *****.com Inc. All rights reserved. ***网  版权所有</div></div></div>
```

由于底部信息很少，而且只是一些数据，所以可以选择使用表格显示这些数据，这样可以使内容的显示更加方便，同时也不影响页面的维护。

20.9.2　定义页面底部的样式

页面底部内容主要包括几个部分，首先是页面底部的背景，然后是各个内容显示的位置，最后是各种内容的颜色。其具体的代码如下所示。

```
#footer {
    clear: both;                                        /*清除底部的漂浮属性*/
    background-image: url(../images/sohu10_10.gif);     /*在页面中添加图片*/
    background-repeat: no-repeat;                       /*背景图片不重复*/
    height: 165px;                          /*定义底部高度为 165 像素*/
}
#innerFooter{
    color: #f4f4f4;                         /*定义字体颜色为亮白色*/
    padding-top: 86px;                      /*顶部补白 86 像素*/
    text-align: center;                     /*定义文字居中*/
}
#innerFooter a{
    color: #f4f4f4;                         /*定义字体颜色为亮白色*/
    text-decoration:none;                   /*定义字体无装饰*/
}
#innerFooter a:hover{
    color:#F5651F;                          /*定义字体颜色为橘红色*/
    text-decoration:underline;              /*定义字体下划线*/
}
#contact {
    margin: 0px auto;                       /*定义上下外边距为 0 像素，左右外边距自动*/
    margin-bottom: 15px;                    /*定义边界底部 15 像素*/
    width: 400px;                           /*定义文本宽度为 400 像素*/
}
#contact td {
    color: #f4f4f4;                         /*定义字体颜色为亮白色*/
    padding: 0 10px;                        /*定义上、下侧补白为 0 像素，左、右侧补白 10 像素*/

}
```

```
#contact a {
    color: #f4f4f4;                        /*定义字体颜色为亮白色*/
    text-decoration: underline;            /*定义字体下划线属性*/
}
#contact a:hover {
    color: #F5651F;                        /*定义字体颜色为橘红色*/
}
```

在上面的代码中首先定义了底部内容的高度，为了能够显示出整个底部使用的背景图片。然后定义了内容显示的颜色和显示的位置。接下来定义了链接内容的显示效果。最后对联系方式的部分进行了进一步修饰。定义页面底部样式后，页面的显示效果如图 20-8 所示。

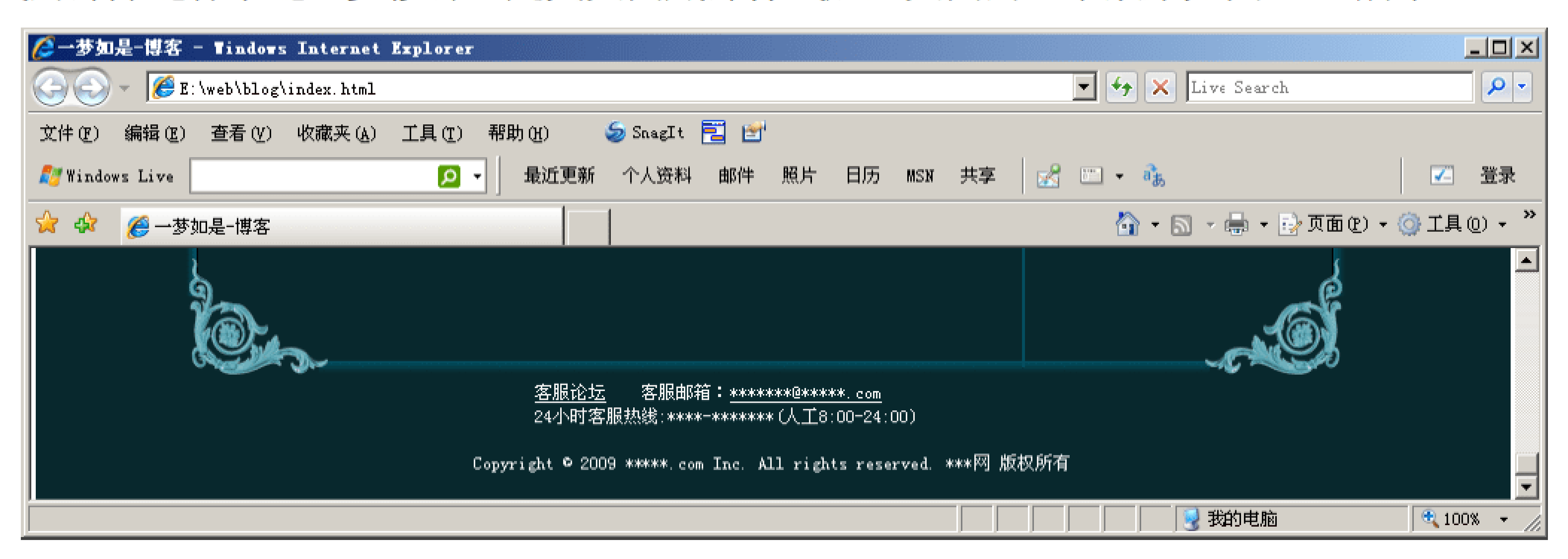


图 20-8　定义页面底部样式后的显示效果